Technical Note 027

Lettura dati PLC e salvataggio valori

Rev 1 - 07/10/2021

Technical Services

Introduzione

Nelle applicazioni SCADA, è spesso necessario elaborare un insieme di variabili I/O al verificarsi di una determinata condizione – per esempio, al completamento di una sequenza di produzione. Comunemente, una simile condizione viene rilevata dal sistema SCADA monitorando una singola variabile I/O. Un PLC può, ad esempio, segnalare la presenza di nuovi dati validi settando a *true* una variabile booleana interna. Questo tipo di eventi può innescare l'esecuzione di uno script lato SCADA per l'elaborazione di altre variabili I/O.

In tali circostanze è possibile che, al momento dell'esecuzione dello script, il driver di comunicazione non abbia ancora letto il valore attuale di tutte le variabili d'interesse, portando così all'elaborazione di dati non aggiornati. Ritardare l'esecuzione dello script può aiutare a mitigare questa problematica, ma non può essere considerata una soluzione deterministica.

La presente nota tecnica illustra due possibili metodi deterministici per assicurare la sincronizzazione di blocchi di attributi I/O nell'ambito di Wonderware System Platform. La trattazione è valida per tutti i driver di comunicazione supportati. Per genericità, non verranno esaminati possibili meccanismi per la gestione di messaggi spontanei (*unsolicited*), come ad es. S7 Block Services.

Soluzione 1: Rilevamento del cambio di valore

Viene qui analizzata una procedura classica, generalmente applicabile anche al di fuori del contesto Wonderware, per attendere l'effettivo aggiornamento di un gruppo di variabili I/O, e seguirà un esempio d'implementazione per System Platform.

Se il dispositivo che produce l'input (es. PLC) sovrascrive *tutte* le variabili d'interesse, *ogni volta* che queste devono essere prelevate dallo SCADA, con un valore *diverso dal precedente*, ne segue che, per essere certi di elaborare l'ultimo aggiornamento, sarà sufficiente effettuare un confronto con l'ultimo valore registrato per ogni variabile letta (opportunamente memorizzato in una seconda variabile ausiliaria dello SCADA), ed attendere così il cambio di valore.

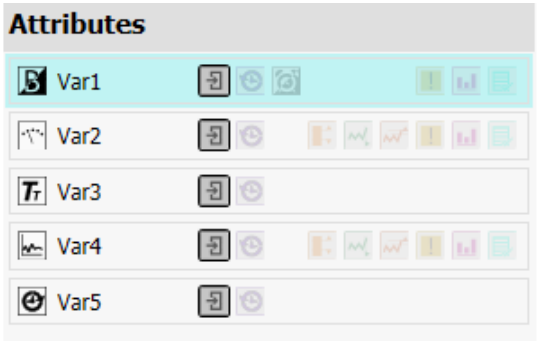
Tuttavia, nel caso piuttosto frequente in cui anche una sola variabile d'interesse non venga necessariamente sovrascritta con un valore diverso dal precedente, l'attesa del cambio di valore non terminerebbe. Per ovviare a questo problema ci si può affidare ad un meccanismo di *timeout*, ma per avere un comportamento deterministico, si può fare in modo che lo SCADA stesso, dopo aver prelevato i dati da processare, sovrascriva le variabili I/O con *valori temporanei*, che *si è certi* verranno modificati dal dispositivo al momento opportuno. Se invece le variabili sono di tipo *read-only* per il sistema SCADA, e si può intervenire sulla logica interna del dispositivo, può essere proprio quest'ultimo a settare i suddetti valori temporanei, che dovranno essere ignorati dallo SCADA dopo la segnalazione dell'avvenuta lettura dei dati validi, fino al successivo cambio di valore.

Nell'esempio che segue, per praticità, si ipotizza una situazione verosimile in cui esiste, per ciascuna variabile d'interesse, un valore speciale che non può essere considerato valido per l'elaborazione, e pertanto può essere usato come valore temporaneo ignorato dal sistema SCADA – ad esempio, si immagina che le variabili numeriche valide non possano assumere il valore 0.
Nella suddetta situazione, la verifica del cambio di valore si semplifica ulteriormente, in quanto non sono necessarie variabili ausiliarie, ma è sufficiente verificare che il valore di ciascuna variabile sia diverso dalla costante di riferimento (es. diverso da 0).

L'implementazione può complicarsi quando non sono facilmente individuabili costanti di riferimento, ovvero quando una variabile può virtualmente assumere tutti i valori possibili ed essere comunque considerata valida. Ciò non vuol dire necessariamente che non vi siano valori temporanei ignorabili dallo SCADA – ad esempio, se è certo che una variabile numerica nel dispositivo può solamente *incrementare* il proprio valore, potrebbe essere usato come valore temporaneo qualsiasi numero *minore* rispetto all'ultimo valore letto.
In molti sistemi informatici, si può risolvere più agevolmente la problematica facendo uso del valore speciale *null*, ma ciò non si applica frequentemente ai sistemi SCADA, che spesso comunicano con dispositivi non in grado di gestire tale valore. Per l'appunto, nel contesto System Platform non è consentito assegnare il valore null ad un attributo di un automation object.

Come esempio, si consideri un automation object dotato degli attributi I/O mostrati in *Figura 1*.

Figura 1: Attributi I/O d'esempio

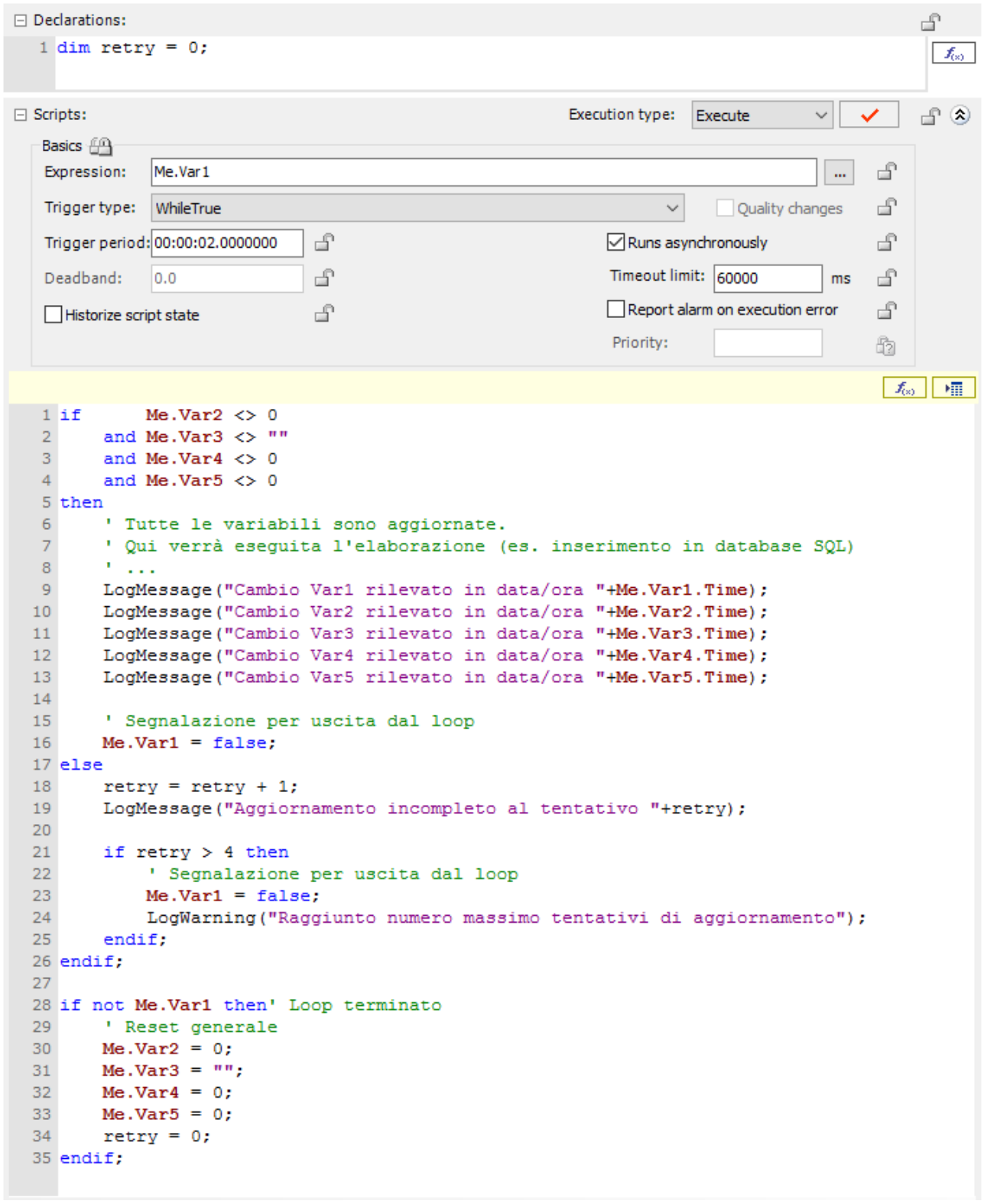


L'implementazione di esempio è mostrata nella successiva *Figura 2*.
La condizione per l'avvio della procedura di elaborazione è che **Var1** sia *true*. Fintanto che tale condizione è verificata, l'oggetto esegue lo script mostrato, con *periodo 2 secondi*.
Si noti che alla riga 21 dello script è valutata una condizione di *timeout*: al quinto tentativo fallito, la procedura viene abortita e viene segnalato l'errore.

Tecnicamente, anziché utilizzare uno script periodico, è anche possibile configurare il Trigger type *OnTrue*, ed usare il metodo *System.Threading.Thread.Sleep()* per implementare l'attesa all'interno dello script eseguito in maniera asincrona.

Nota: la proprietà **.Time** di un attributo fornisce il *timestamp dell'ultimo cambio di valore* registrato (un esempio di utilizzo è visibile alle righe 9-13 dello script in *Figura 2*). Tale proprietà è certamente utile, ma difficilmente sfruttabile per risolvere la problematica iniziale. Nell'esempio specifico, non sarebbe corretto attendere che il timestamp degli attributi d'interesse sia successivo a Var1.Time, in quanto, per l'appunto, non è noto l'ordine di lettura degli I/O.

Figura 2: Esempio implementazione soluzione 1



Va precisato infine che, nel caso qui considerato, in cui è il sistema SCADA ad occuparsi del reset delle variabili, è importante che sia lo stesso SCADA a prendere in carico l'*inizializzazione* dei valori all'avvio (*fase non mostrata nell'esempio*), in modo da evitare possibili anomalie in esercizio.

Soluzione 2: OPCClient Block Read

A differenza del DeviceIntegration (DI) object DDESuiteLinkClient, l'oggetto OPCClient è in grado di gestire *transazioni* per la sincronizzazione di gruppi di variabili I/O (Block Reads).
Per ogni OPCClient, è possibile creare diversi blocchi di lettura, ognuno dei quali potrà gestire una transazione per volta, tramite specifiche proprietà.

Una transazione viene avviata settando a true la proprietà *TransactionTrigger*. Quest'azione forza la lettura del blocco di variabili da parte del driver di comunicazione legato all'oggetto OPCClient.
Solo quando tutte le variabili del blocco sono aggiornate, la transazione ha termine e la proprietà *TransactionComplete* viene settata a true per segnalare il completamento dell'operazione.

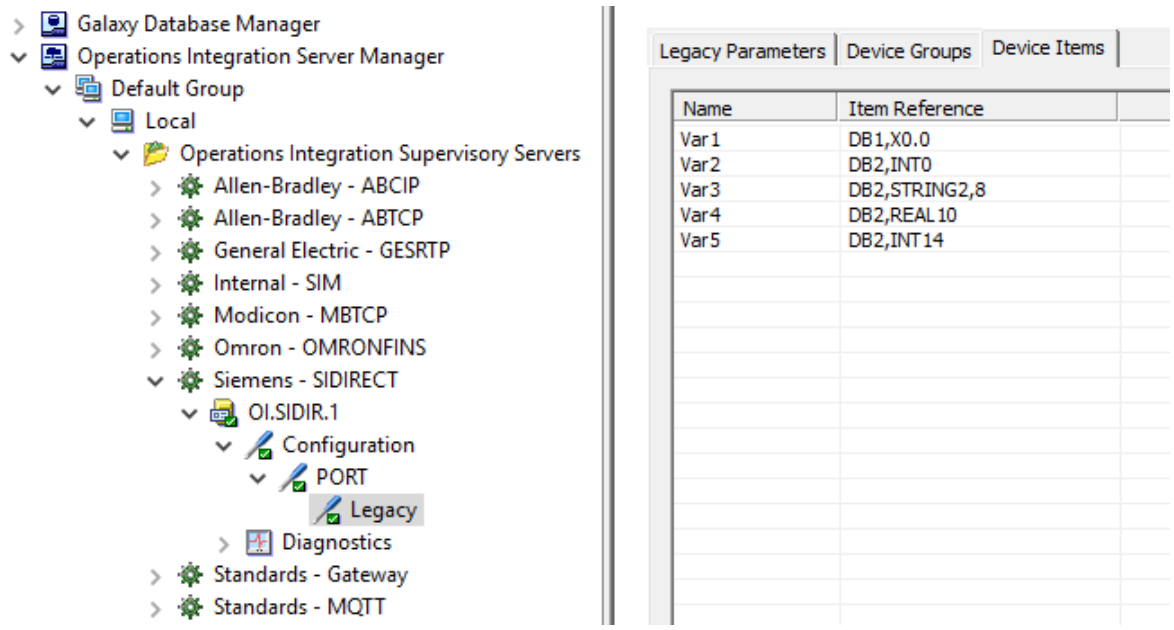
Nota: la proprietà .TransactionComplete non viene resettata automaticamente, per ovvi motivi.

Tra le altre, le proprietà *TransactionStart* e *TransactionEnd* possono essere usate per ottenere i timestamp di inizio e fine dell'ultima transazione registrata.

L'oggetto OPCClient può essere usato parallelamente all'oggetto DDESuiteLinkClient, in quanto ogni OI Server espone i dati sia in protocollo SuiteLink che in protocollo OPC.
Rispetto alla soluzione 1, la funzionalità Block Read, oltre alla maggiore semplicità, offre il vantaggio di poter leggere gruppi di variabili on-demand, se il polling per tali variabili non è necessario.

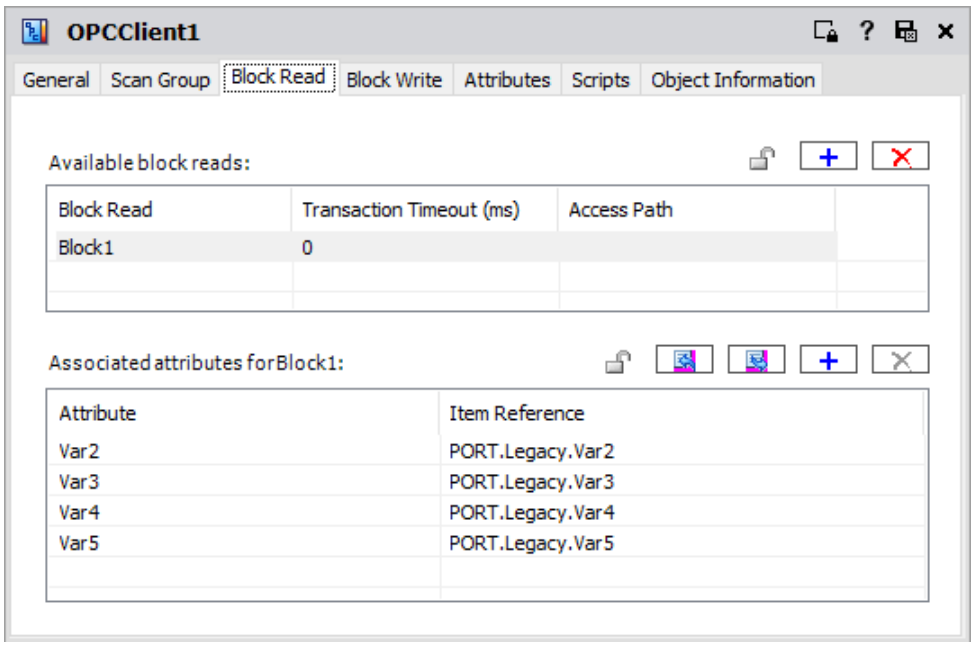
Si consideri l'esempio di configurazione driver riportato in *Figura 3*.
Supponiamo che l'attributo Var1 sia gestito mediante oggetto DDESuiteLinkClient.

Figura 3: Esempio configurazione driver di comunicazione



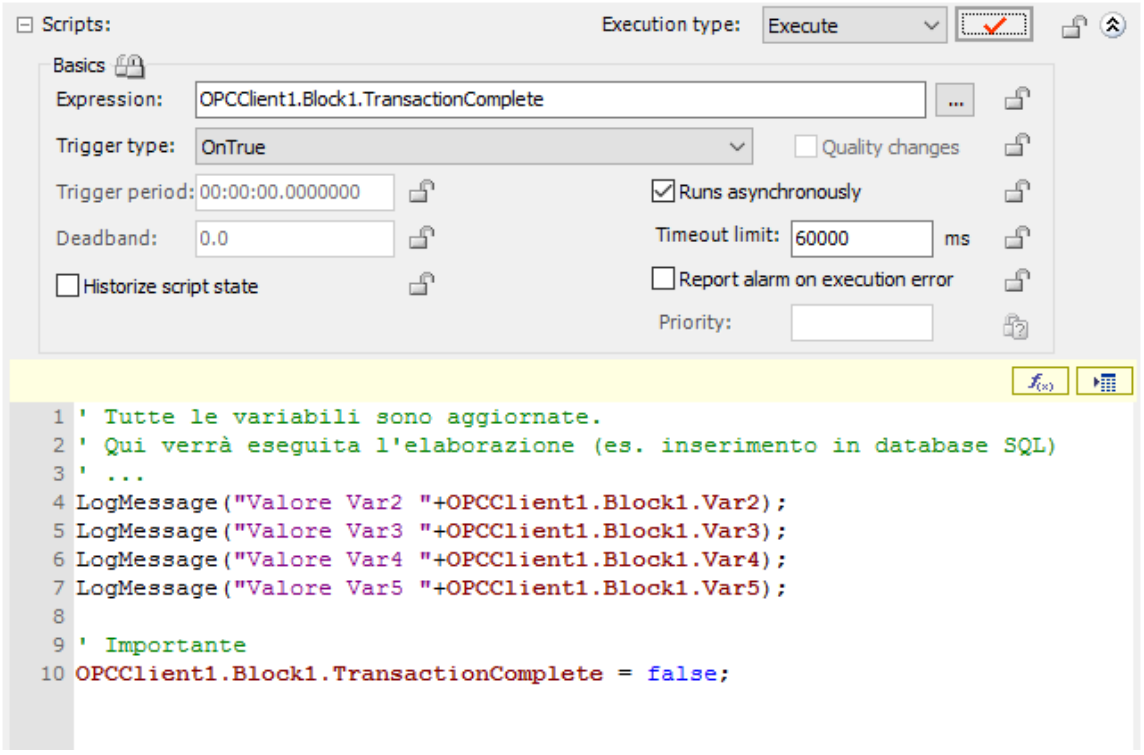
In tale esempio, l'oggetto OPCClient avrà Server name: OI.SIDIR.1 (tab General).
La configurazione Block Read sarà simile a quella mostrata nella *Figura 4* seguente.
Si noti che in Item Reference è anche possibile utilizzare direttamente gli indirizzi del driver.
È possibile specificare un timeout per la transazione. Lo stato dell'ultima transazione è ispezionabile mediante proprietà *TransactionStatus*.
Pur non essendo necessario, è opportuno aggiungere uno scan group, anche inesistente, in modo da evitare il warning in fase di salvataggio dell'oggetto OPCClient (tab Scan Group).

Figura 4: Esempio configurazione oggetto OPCClient



La soluzione prevede due script. Il primo script dovrà semplicemente settare a true la proprietà OPCClient1.Block1.TransactionTrigger al cambio dell'attributo Var1. Il secondo script è mostrato in *Figura 5*. Si noti che i riferimenti agli attributi del blocco di lettura (qui referenziati direttamente alle righe 4-7) possono essere assegnati come Input Source di attributi I/O in altri automation objects.

Figura 5: Esempio implementazione soluzione 2



Autore: M. Popolla

Disclaimer

Il presente documento è fornito a scopo di esempio e non sostituisce la documentazione AVEVA. L'applicazione di quanto contenuto, in un preciso ambito applicativo, deve essere sempre validata da un tecnico Wonderware. La documentazione rilasciata da AVEVA resta il riferimento tecnico ufficiale da seguire: softwaresupport.aveva.com. Wonderware Italia non si assume la responsabilità di un'applicazione scorretta di questo documento.