



- ▶ Version 10.1
- ▶ Synchronising Threads
- ▶ Introduction

Synchronising Threads – Token Pool

Dyalog Version 10.1 provides a comprehensive mechanism for synchronising threads. This is done using a token pool.

An application can synchronise its threads by having one thread add tokens into the pool whilst other threads wait for tokens to become available and retrieve them from the pool.

Tokens possess two separate attributes, a type and a value.

- The type of a token is a positive or negative integer scalar.
- The value of a token is any arbitrary array that you might wish to associate with it.

The token pool may contain up to 2^{*31} tokens; they do not have to be unique either in terms of their types or of their values.

The following system functions are used to manage the token pool:

<code>□TPUT</code>	Puts Tokens into the pool.	
<code>□TGET</code>	If necessary waits for, and then gets some tokens from the pool.	
<code>□TPOOL</code>	Reports the types of tokens in the pool.	
<code>□TREQ</code>	Reports the token request from specific threads.	

A simple example of a thread synchronisation requirement occurs when you want one thread to reach a certain point in processing before a second thread can continue. Perhaps the first thread performs a calculation, and the second thread must wait until the result is available before it can be used.

This can be achieved by having the first thread put a specific type of token into the pool using `□TPUT`. The second thread waits (if necessary) for the new value to be available by calling `□TGET` with the same token type.

Notice that when `□TGET` returns, the specified tokens are removed from the pool. However, negative token types will satisfy an infinite number of requests for their positive equivalents.

Dyalog Ltd

South Barn
Minchens Court
Minchens Lane
Bramley
Hampshire
RG26 5BH
United Kingdom

Phone:

+ 44 (0) 1256 830 030

Fax:

+ 44 (0) 1256 830 031

e-mail:

sales@dyalog.com



Version 10.1

Synchronising Threads

Introduction

Examples

`□TGET □TPOOL` *empty the pool.*

`□TPUT 2 3 2` *put a 2-token, a 3-token and another 2-token into the pool.*

`□TPUT 2` *put another 2-token into the pool*

`□TPOOL` *tokens in pool.*
2 3 2 2

`□TGET 3 2` *remove a 3-token and a 2-token from the pool.*

`□TPOOL`
2 2

`0.5 □TGET 2` *wait up to half a second for a 2-token.*

`□TGET □TPOOLn2` *remove all 2-tokens from the pool.*

The system is designed to cater for more complex forms of synchronisation. For example, a **semaphore** to control a number of resources can be implemented by keeping that number of tokens in the pool. Each thread will take a token while processing, and return it to the pool when it has finished.

A second complex example is that of a **latch** which holds back a number of threads until the coast is clear. At a signal from another thread, the latch is opened so that all of the threads are released. The latch may (or may not) then be closed again to hold up subsequently arriving threads. A practical example of a latch is a ferry terminal.

Dyalog Ltd

South Barn
Minchens Court
Minchens Lane
Bramley
Hampshire
RG26 5BH
United Kingdom

Phone:

+ 44 (0) 1256 830 030

Fax:

+ 44 (0) 1256 830 031

e-mail:

sales@dyalog.com