# Example24

# CCR Interop with Non-CCR Code

## Example 24: Coordination with main application thread

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Text;
using Microsoft.Ccr.Core;
using System.Threading;
using System.Net;
using Microsoft.Ccr.Adapters.IO;

namespace Example
{
    class Program
    {
        // create dispatcher and dispatcher queue for scheduling tasks
        static Dispatcher dispatcher = new Dispatcher();
        static DispatcherQueue _taskQueue = new DispatcherQueue("sample queue", dispatcher);

        //Example 25: Coordination with main application thread
        static void Main(string[] args)
        {
            /// <summary>
            /// 放值的方式
            /// use a System.Threading.AutoResetEvent and block the main application thread until the
CCR application is finished
            /// </summary>

            // create OS event used for signalling
            // AutoResetEvent: Notifies a waiting thread that an event has occurred. This class cannot
be inherited.
            AutoResetEvent signal = new AutoResetEvent(false);

            // schedule a CCR task that will execute in parallel with the rest of
            // this method
            Arbiter.Activate(
                _taskQueue,
                new Task<AutoResetEvent>(signal, SomeTask)
            );

```

```
            ThrottlingExample();
            // block main application thread form exiting
            // WaitOne: When overridden in a derived class, blocks the current thread until the current
WaitHandle receives a signal.
            Console.WriteLine("Before WaitOne.");
            signal.WaitOne();
            Console.WriteLine("After WaitOne.");
        }

        static void ThrottlingExample()
        {
            int maximumDepth = 10;
            Dispatcher dispatcher = new Dispatcher(0, "throttling example");
            DispatcherQueue depthThrottledQueue = new DispatcherQueue("ConstrainQueueDepthDiscard",
                  dispatcher,
                  TaskExecutionPolicy.ConstrainQueueDepthDiscardTasks,
                  maximumDepth);

            Port<int> intPort = new Port<int>();
            Arbiter.Activate(depthThrottledQueue,
               Arbiter.Receive(true, intPort,
               delegate(int i)
               {
                   // only some items will be received since throttling will discard most of them
                   Console.WriteLine(i);
               })
            );

            // post items as fast as possible so that the depth policy is activated and discards
            // all the oldest items in the dispatcher queue
            for (int i = 0; i < maximumDepth * 10; i++)  // * 10000000
            {
                intPort.Post(i);
            }
        }

        /// <summary>
        /// Handler that executes in parallel with main application thread
        /// </summary>
        /// <param name="signal"></param>
        static void SomeTask(AutoResetEvent signal)
        {
            try
            {
                for (int i = 0; i < 100; i++)
                {
                    int k = -i;
                    Console.WriteLine(k);
                }
            }
            finally
            {
```

```
95                  // done, signal main application thread
96                  signal.Set();
97              }
98          }
99      }
100 }
```