Got it — in **Step 2** you must copy **both** the .c **and** the .h files (src and include), because a fresh Visual Studio project doesn’t know where to find ThreadX headers unless you give them.

Let’s go through the **full steps again**, with that clarified and cleaned up so you have an exact checklist.

**Beginner Guide — Creating a New Visual Studio Project for ThreadX (Win32 Port)**

**Step 1 — Create a new project**

1. Open **Visual Studio**.
2. Go to **File → New → Project**.
3. Search for **Console App** → choose **Console App** under C++ → click **Next**.
4. Name it something like:

nginx

CopyEdit

ThreadXPlayground

and choose a location **outside** your cloned ThreadX repo.

1. Click **Create**.

**Step 2 — Get ThreadX Win32 source and header files**

You’ll need **both source (.c)** and **header (.h)** files from two locations in your cloned repo:

1. **Core kernel (common)**
   * .c files:

bash

CopyEdit

threadx/common/src

* + .h files:

bash

CopyEdit

threadx/common/inc

1. **Win32 port for MS Visual Studio**
   * .c files:

bash

CopyEdit

threadx/ports/win32/ms\_visual\_studio/src

* + .h files:

bash

CopyEdit

threadx/ports/win32/ms\_visual\_studio/inc

1. In your Visual Studio project folder, create a new folder called:

nginx

CopyEdit

threadx

Inside it, you can make two subfolders if you want to keep things tidy:

bash

CopyEdit

threadx/src

threadx/inc

1. Copy:
   * **All .c files** from both common/src and ports/win32/ms\_visual\_studio/src into threadx/src.
   * **All .h files** from both common/inc and ports/win32/ms\_visual\_studio/inc into threadx/inc.

**Step 3 — Add ThreadX files to your Visual Studio project**

1. In Solution Explorer, right-click your project → **Add → Existing Item…**.
2. Navigate to threadx/src and select all .c files → click **Add**.
3. Headers don’t need to be explicitly added to the project, but they must be in the include path.

**Step 4 — Configure include paths**

1. Right-click your project → **Properties**.
2. Go to **C/C++ → General → Additional Include Directories**.
3. Add the path to your headers:

javascript

CopyEdit

$(ProjectDir)threadx\inc

1. Click **OK**.

**Step 5 — Link Winmm.lib**

1. Right-click your project → **Properties**.
2. Go to **Linker → Input → Additional Dependencies**.
3. Add:

vbnet

CopyEdit

Winmm.lib

1. Click **OK**.

**Step 6 — Replace the default code with a ThreadX example**

1. Delete the default main.cpp that Visual Studio created.
2. Add a new file called main.c.
3. Paste this code:

c

CopyEdit

#include <stdio.h>

#include "tx\_api.h"

#define THREAD\_STACK\_SIZE 1024

#define QUEUE\_MESSAGE\_COUNT 8

TX\_THREAD sender\_thread;

TX\_THREAD receiver\_thread;

ULONG sender\_stack[THREAD\_STACK\_SIZE / sizeof(ULONG)];

ULONG receiver\_stack[THREAD\_STACK\_SIZE / sizeof(ULONG)];

TX\_QUEUE my\_queue;

ULONG queue\_area[QUEUE\_MESSAGE\_COUNT];

void sender\_entry(ULONG thread\_input)

{

(void)thread\_input;

ULONG value = 0;

while (1)

{

value++;

tx\_queue\_send(&my\_queue, &value, TX\_NO\_WAIT);

tx\_thread\_sleep(50);

}

}

void receiver\_entry(ULONG thread\_input)

{

(void)thread\_input;

ULONG recv\_value;

while (1)

{

if (tx\_queue\_receive(&my\_queue, &recv\_value, TX\_WAIT\_FOREVER) == TX\_SUCCESS)

{

printf("Received: %lu\n", recv\_value);

}

}

}

void tx\_application\_define(void \*first\_unused\_memory)

{

tx\_queue\_create(&my\_queue, "MyQueue", sizeof(ULONG),

queue\_area, sizeof(queue\_area));

tx\_thread\_create(&receiver\_thread, "Receiver", receiver\_entry, 0,

receiver\_stack, sizeof(receiver\_stack),

1, 1, TX\_NO\_TIME\_SLICE, TX\_AUTO\_START);

tx\_thread\_create(&sender\_thread, "Sender", sender\_entry, 0,

sender\_stack, sizeof(sender\_stack),

2, 2, TX\_NO\_TIME\_SLICE, TX\_AUTO\_START);

}

int main(void)

{

tx\_kernel\_enter();

return 0;

}

**Step 7 — Build and run**

1. Press **Ctrl+Shift+B** to build.
2. If there are no errors, press **F5** to run.
3. You should see:

makefile

CopyEdit

Received: 1

Received: 2

Received: 3

...

**Step 8 — Experiment**

* Change tx\_thread\_sleep() values to see how it affects message rates.
* Add another sender to push different data into the queue.
* Change priorities (1 is higher priority than 2) and see preemption.

If you want, I can make you a **ready-to-open Visual Studio ZIP** with all the .c and .h files already added so you can skip file-copying and go straight to F5.