Write a self-contained function block in 61131-3 structured text that implements a timer.

The IEC 61131-3 standard is used to create industrial automation software, and structured text is one of its programming languages. Here's a simple timer function block written in 61131-3 structured text:

sql Copy code

```
FUNCTION_BLOCK Timer
VAR_INPUT
  Start: BOOL; // Start the timer
  Reset: BOOL; // Reset the timer
  PresetTime: TIME; // The time to count
END_VAR

VAR_OUTPUT
  ElapsedTime: TIME; // Time elapsed since the timer started
  Done: BOOL; // Indicates if the timer has completed
END_VAR

VAR
  StartTime: TIME;
  Running: BOOL;
END_VAR

METHOD RunTimer: BOOL
  Running := Start AND NOT Done;
  RETURN Running;
END_METHOD

METHOD ResetTimer: BOOL
  StartTime := 0;
  ElapsedTime := 0;
  Done := FALSE;
  RETURN Reset;
END_METHOD

METHOD UpdateElapsedTime: BOOL
  ElapsedTime := TIME() - StartTime;
  RETURN ElapsedTime >= PresetTime;
END_METHOD

METHOD SetDone: BOOL
  Done := TRUE;
  RETURN Done;
END_METHOD
END_FUNCTION_BLOCK
```

To use the timer, create an instance of the Timer function block, and call its methods in your main program. Here's an example of how to do that:

css Copy code

```
PROGRAM Main
VAR
  myTimer: Timer;
  startSignal: BOOL := FALSE;
  resetSignal: BOOL := FALSE;
END_VAR

IF startSignal THEN
  myTimer.Start := TRUE;
  myTimer.Reset := FALSE;
ELSIF resetSignal THEN
  myTimer.Start := FALSE;
  myTimer.Reset := TRUE;
ELSE
  myTimer.Start := FALSE;
  myTimer.Reset := FALSE;
END_IF

IF myTimer.RunTimer() THEN
  IF NOT myTimer.Running THEN
    myTimer.StartTime := TIME();
  END_IF;

  IF myTimer.UpdateElapsedTime() THEN
    myTimer.SetDone();
  END_IF;
ELSIF myTimer.ResetTimer() THEN
  // Timer has been reset
END_IF;
END_PROGRAM
```

The main program starts or resets the timer based on the `**startSignal**` and `**resetSignal**` boolean variables, which can be controlled by external inputs or other logic within your program.