Program Listing for File event_based_scheduler.hpp

```cpp
/* * SPDX-FileCopyrightText: Copyright (c) 2024 NVIDIA CORPORATION & AFFILIATES.
All rights reserved. * SPDX-License-Identifier: Apache-2.0 * * Licensed under the
Apache License, Version 2.0 (the "License"); * you may not use this file except in
compliance with the License. * You may obtain a copy of the License at * *
http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law
or agreed to in writing, software * distributed under the License is distributed on an
"AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied. * See the License for the specific language governing
permissions and * limitations under the License. */ #ifndef
HOLOSCAN_CORE_SCHEDULERS_GXF_EVENT_BASED_SCHEDULER_HPP #define
HOLOSCAN_CORE_SCHEDULERS_GXF_EVENT_BASED_SCHEDULER_HPP #include
<cstdint> #include <memory> #include <string> #include
<gxf/std/event_based_scheduler.hpp> #include "../../gxf/gxf_scheduler.hpp"
#include "../../resources/gxf/clock.hpp" #include
"../../resources/gxf/realtime_clock.hpp" namespace holoscan { class
EventBasedScheduler : public gxf::GXFScheduler { public:
HOLOSCAN_SCHEDULER_FORWARD_ARGS_SUPER(EventBasedScheduler,
gxf::GXFScheduler) EventBasedScheduler() = default; const char* gxf_typename()
const override { return "nvidia::gxf::EventBasedScheduler"; } std::shared_ptr<Clock>
clock() override { return clock_.get(); } void setup(ComponentSpec& spec) override;
void initialize() override; // Parameter getters used for printing scheduler description
(e.g. for Python __repr__) int64_t worker_thread_number() { return
worker_thread_number_; } bool stop_on_deadlock() { return stop_on_deadlock_; }
int64_t stop_on_deadlock_timeout() { return stop_on_deadlock_timeout_; } // could
return std::optional<int64_t>, but just using int64_t simplifies the Python bindings
int64_t max_duration_ms() { return max_duration_ms_.has_value() ?
max_duration_ms_.get() : -1; } nvidia::gxf::EventBasedScheduler* get() const; private:
Parameter<std::shared_ptr<Clock>> clock_; Parameter<int64_t>
worker_thread_number_; Parameter<bool> stop_on_deadlock_; Parameter<int64_t>
max_duration_ms_; Parameter<int64_t> stop_on_deadlock_timeout_; // in ms // The
following parameter needs to wait on ThreadPool support // Parameter<bool>
```

thread_pool_allocation_auto_; }; } *// namespace holoscan* #endif/*
HOLOSCAN_CORE_SCHEDULERS_GXF_EVENT_BASED_SCHEDULER_HPP */