Program Listing for File clock.hpp

Return to documentation for file (include/holoscan/core/resources/gxf/clock.hpp)

/* * SPDX-FileCopyrightText: Copyright (c) 2023-2024 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_RESOURCES_GXF_CLOCK_HPP #define HOLOSCAN_CORE_RESOURCES_GXF_CLOCK_HPP #include <string> #include <gxf/std/clock.hpp> #include "../../gxf/gxf_resource.hpp" namespace holoscan { class Clock : public gxf::GXFResource { public: HOLOSCAN_RESOURCE_FORWARD_ARGS_SUPER(Clock, gxf::GXFResource) Clock() = default; Clock(const std::string& name, nvidia::gxf::Clock* component); const char* gxf_typename() const override { return "nvidia::gxf::Clock"; } virtual double time() const = 0; virtual int64_t timestamp() const = 0; virtual void sleep_for(int64_t duration_ns) = 0; template <typename Rep, typename Period> void sleep_for(std::chrono::duration<Rep, Period> duration) { int64_t duration_ns = std::chrono::duration_cast<std::chrono::nanoseconds>(duration).count(); sleep_for(duration_ns); } virtual void sleep_until(int64_t target_time_ns) = 0; nvidia::gxf::Clock* get() const; }; } *// namespace holoscan* #endif/* HOLOSCAN_CORE_RESOURCES_GXF_CLOCK_HPP */