Program Listing for File ntv2channel.hpp

Return to documentation for file (`include/holoscan/operators/aja_source/ntv2channel.hpp`)

```
/* * SPDX-FileCopyrightText: Copyright (c) 2022-2023 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_OPERATORS_AJA_SOURCE_NTV2CHANNEL_HPP #define
HOLOSCAN_OPERATORS_AJA_SOURCE_NTV2CHANNEL_HPP #include
<ajantv2/includes/ntv2enums.h> #include <yaml-cpp/yaml.h> #include <sstream>
#include <string> template <> struct YAML::convert<NTV2Channel> { static Node
encode(const NTV2Channel& rhs) { Node node; auto channel = static_cast<int>(rhs)
+ 1; // 0 => NTV2_CHANNEL1 std::stringstream ss; ss << "NTV2_CHANNEL"; ss <<
channel; node.push_back(ss.str()); YAML::Node value_node = node[0]; return
value_node; } static bool decode(const Node& node, NTV2Channel& rhs) { if
(!node.IsScalar()) return false; const std::string prefix("NTV2_CHANNEL"); auto value
= node.Scalar(); if (value.find(prefix) != 0) return false; value =
value.substr(prefix.length()); try { size_t len; const auto index = std::stoi(value, &len);
if (index < 1 || index > NTV2_MAX_NUM_CHANNELS || len != value.length()) { return
false; } rhs = static_cast<NTV2Channel>(index - 1); return true; } catch (...) { return
false; } } }; #endif/* HOLOSCAN_OPERATORS_AJA_SOURCE_NTV2CHANNEL_HPP */
```