NDS v2.3. Release notes

Table of Contents

1 INTERFACE CHANGES

1.1 NDS Library Renamed

The NDS library was renamed from `libndsCPP.so` to `libnds.so`.

All links to the NDs library must be fixed.

1.2 Device State Machine Changes

Now the device goes to the INIT state with the ON message. If required, immediately switch the device to the ON state (fast init), then the following code should be used:

```
ExDevice::ExDevice(const std::string& name)
{
…
      enableFastInit();
…
}
```

This call will register the onEnter handler for the INIT state, which will request the ON state immediately after entering the INIT state.

1.3 ChannelGroup and Channel State Machine Changes

NDS v.2.3 changes the state machine for ChannelsGroup and Channel.

The state machine which existed in NDS v.2.2.5 is depicted in Figure 1.

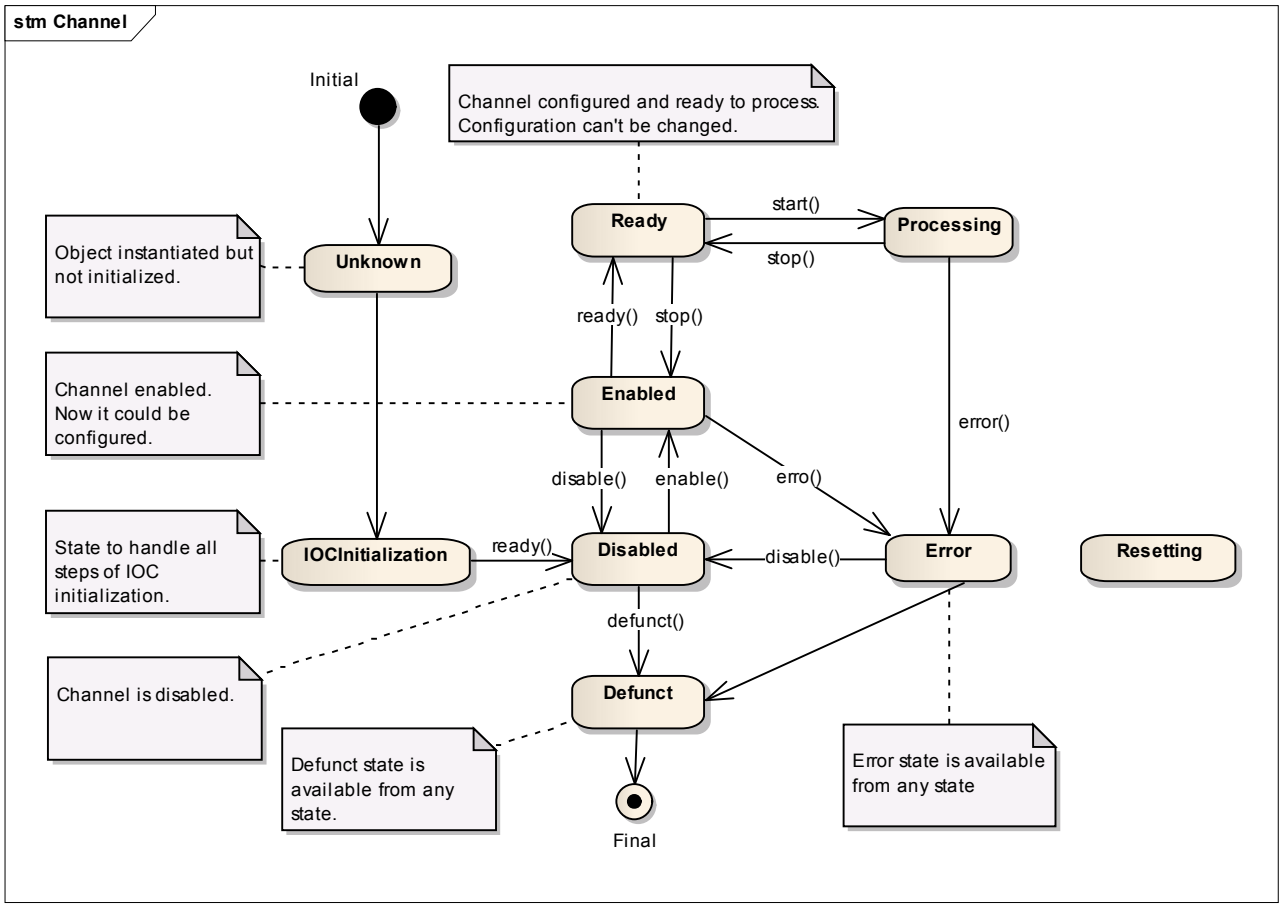

Figure 1 NDS 2.2.5 ChannelGroup's and Channel's state machine

NDS v.2.3 has a reworked state machine. The states READY and ENABLED are deprecated. READY and PROCESSING represented 2 sub-states of a single state: PROCESSING. It makes handling of these 2 states complicated and from a development perspective could cause a number of mistakes. The new state machine will only have the PROCESSING state. The ENABLED state plays the role of a flag indicating whether an object is enabled or not. To make the model consistent, if a parent object goes into the DISABLED state all children objects (in relationship ChannelGroup - Channel) should also go into the OFF state. After that it is impossible to restore configuration of child objects. The new model only has a DISABLED state and introduces the PV (-ENBL) which indicates whether the object should be enabled or not on initialization. Here "enabled" means that the object is configured and ready to be used. Each object could be enabled independently from its parent. It means that a disabled ChannelGroup could have enabled Channels. This solution simplifies state handling and allows preservation of an object's state between device restarts. The new state machine is depicted in Figure 2.

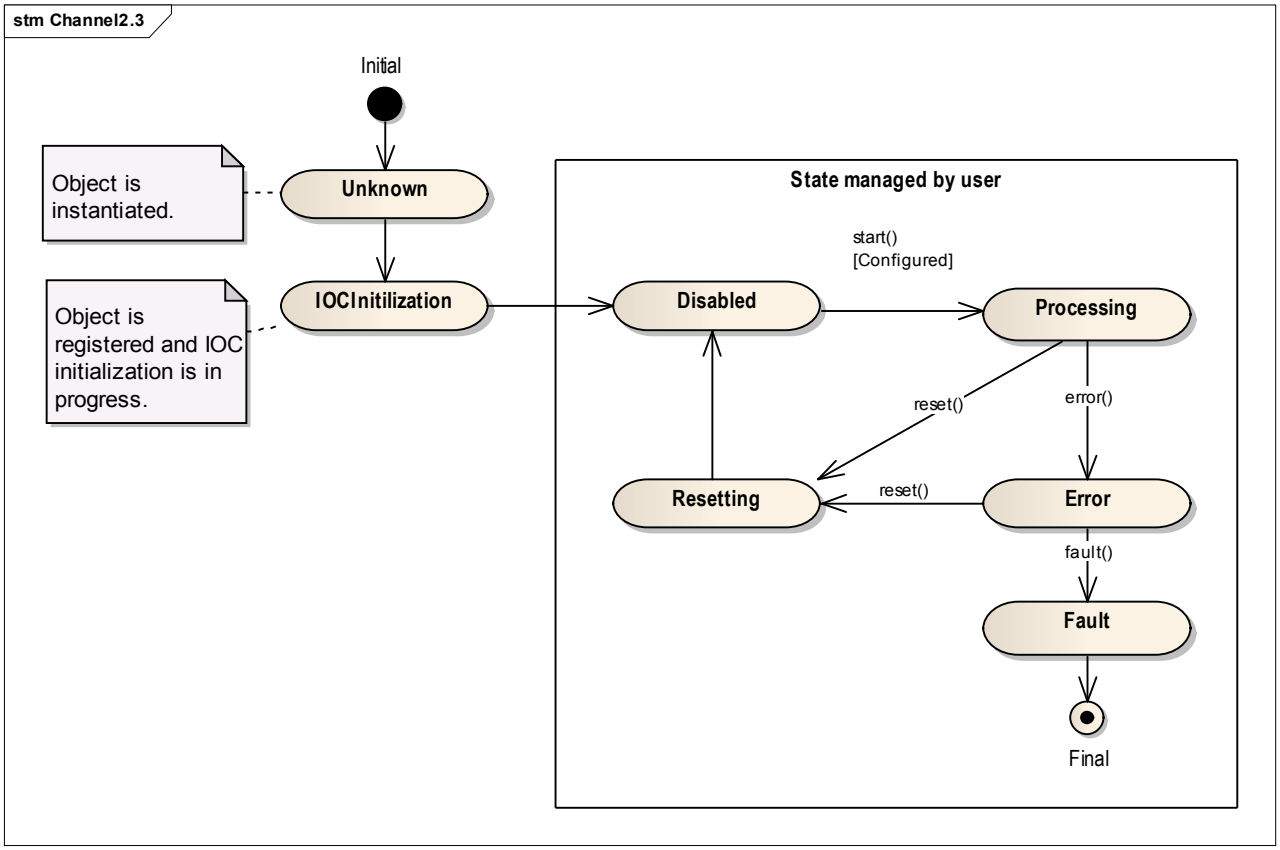

Figure 2 NDS v.2.3 ChannelGroup's and Channel's state machine

The ERROR and FAULT states are accessible from any other state (is not shown in Figure 2).

The transition from the RESETTING state is managed by the developer by use of special requests (is not shown in Figure 2).

NDS v.2.3 will support the obsolete states until the next version. All functions which will be removed in future versions are marked as "deprecated". Please ensure that you don't get warnings during compilation.

1.3.1 Auto Start

Introduction of the ENABLED/DISABLED flag allows introduction of an auto start procedure. It means that an object which has the **–ENBL** PV in state 1 (Enabled) will be automatically switched to the PROCESSING state immediately after IOC initialization is complete.

It provides the ability to configure the initial device state from configuration tools, like SDD.

2 NEW FEATURES

2.1 NDS Tasks

NDS introduces tasks to serve threading needs. The new tasks are:

- ndsThreadTask
- ndsPeriodicTask
- ndsPollingTask

From previous versions the following task also exists:

- nds:Timer.

All tasks have a common interface and each new task is registered within nds::TaskManager.

- ThreadTask is a simple thread wrapper which provides an easy way to register callback to be run in a separate thread.
- PeriodicTask allows running of callback with requested period.
- PollingTask provides a core for file polling. It uses epoll functions.
- Timer is a wrapper over epicsTimer to provide a common task interface.

Users can manage tasks through the IOC console with the following functions:

- ndsListTasks
- ndsStartTask
- ndsCancelTask

For details see the NDS developer's manual. For examples see EPICS application templates provided by NDS.