

Title: The future prospects of solar energy storage field

Generated on: 2026-01-31 04:12:00

Copyright (C) 2026 SMART SYSTEMS S.L. All rights reserved.

If the future is the result of a call to `std::async` that used lazy evaluation, this function returns immediately without waiting. This function may block for longer than ...

If the future is the result of a call to `async` that used lazy evaluation, this function returns immediately without waiting. The behavior is undefined if `valid ()` is false before the call ...

The promise is the "push" end of the promise-future communication channel: the operation that stores a value in the shared state synchronizes-with (as defined in ...

```
future (const future & ) = delete; ~future (); future & operator =(const future & ) = delete; future & operator
=(future & & ) noexcept; shared_future &lt;R>; share () noexcept; // ...
```

The `get` member function waits (by calling `wait ()`) until the shared state is ready, then retrieves the value stored in the shared state (if any). Right after calling this function, `valid ...`

A future represents the result of an asynchronous operation, and can have two states: uncompleted or completed. Most likely, as you aren't doing this just for fun, you actually ...

The class template `std::future` provides a mechanism to access the result of asynchronous operations: An asynchronous operation (created via `std::async`, ...

Specifies state of a future as returned by `wait_for` and `wait_until` functions of `std::future` and `std::shared_future`. Constants

Checks if the future refers to a shared state. This is the case only for futures that were not default-constructed or moved from (i.e. returned by `std::promise::get_future ()`, ...

Website: <https://www.smart-telecaster.es>

The future prospects of solar energy storage field

Source: <https://www.smart-telecaster.es/Thu-09-Nov-2023-26957.html>

Website: <https://www.smart-telecaster.es>

