Started

exit/ DeInitHal(), DeInitSpi()

Idle

entry/ retryCnt = 0

IdleNormal

entry/ InitSpi(), InitHal(),
ES_WIFI_RegisterBusIO(),
if ((ES_WIFI_Init() != OK) ||
(ES_WIFI_GetMACAddress(m_macAddr) != OK))
Raise(INIT_FAILED)

RETRY_TIMER

INIT_FAILED

IdleRetry

entry/ DeInitHal(), DeInitSpi(),
start retryTimer,
if (++retry > 2) Raise(FAULT_EVT)
exit/ stop retryTimer, recall
WIFI_CONNECT_REQ, WIFI_DISCONNECT_REQ/ defer

Fault

TBD

FAULT_EVT

WIFI_CONNECT_REQ
/ m_inEvt = req,
save domain to m_domain,
save port to m_port,
save FIFOs to m_dataOutFifo, m_dataInFifo

DISCONNECTED
/ Send(WIFI_DISCONNECT_IND, m_client)

DisconnectWait is not needed since entry to Idle will reset the WiFi module.

DONE

Running

exit/ clear m_dataOutFifo and m_dataInFifo,
DeInitHal(), DeInitSpi()

ConnectWait

entry/ start stateTimer
exit/ stop stateTimer

Joining

entry/ if (ES_WIFI_ListAccessPoints() == OK)
if (wifiSsid found in list)
if (ES_WIFI_Connect(wifiSsid, wifiPwd) == OK)
Send(DONE, self)
else Raise(FAILED)
else start retryTimer
else Raise(FAILED)
exit/ stop retryTimer

It "sends" DONE instead of "raises" to allow any DISCONNECT or STOP requests to be processed first.

RETRY_TIMER

DONE

Connecting

entry/ if (ES_WIFI_StartClientConnection() == OK)
Raise(DONE)
else Raise(FAILED)

FAILED, STATE_TIMER
/ SendCfm(WIFI_CONNECT_CFM(error), m_inEvt)

DisconnectWait

exit/ recall
WIFI_DISCONNECT_REQ, WIFI_STOP_REQ/ defer

The calls ES_WIFI_Disconnect() and ES_WIFI_StopClientConnection() are commented as they don't work after a previous timeout error. This state is a placeholder.

Unjoining

entry/ /*ES_WIFI_Disconnect(),*/
start stateTimer, Raise(DONE)
exit/ stop stateTimer
STATE_TIMER/ Raise(DONE)

DONE

Disconnecting

entry/ /*ES_WIFI_StopClientConnection(),*/
start stateTimer, Raise(DONE)
exit/ stop stateTimer
STATE_TIMER/ Raise(DONE)

WIFI_DISCONNECT_REQ, WIFI_STOP_REQ
/ defer

DONE
/ SendCfm(WIFI_CONNECT_CFM(SUCCESS), m_inEvt)

Connected