

VINETIC[®]

Voice and Internet Enhanced Telephony Interface
Circuit

T.38 Fax Agent Release 1.1

Programmer's Reference

CONFIDENTIAL
Distribution with NDA only

Communications



N e v e r s t o p t h i n k i n g .

Edition 2005-09-22

**Published by Infineon Technologies AG,
St.-Martin-Strasse 53,
81669 München, Germany**

**© Infineon Technologies AG 2005.
All Rights Reserved.**

Attention please!

The information herein is given to describe certain components and shall not be considered as a guarantee of characteristics.

Terms of delivery and rights to technical change reserved.

We hereby disclaim any and all warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (www.infineon.com).

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

CONFIDENTIAL

T.38 FAX Agent Release 1.1, User's Manual Programmer's Reference**CONFIDENTIAL****Revision History: 2005-09-22, Rev. 1.0****Previous Version: none**

Page	Subjects (major changes since last revision)

Trademarks

ABM[®], ACE[®], AOP[®], ARCOFI[®], ASM[®], ASP[®], DigiTape[®], DuSLIC[®], EPIC[®], ELIC[®], FALC[®], GEMINAX[®], IDEC[®], ATA[®], IOM[®], IPAT[®]-2, ISAC[®], ITAC[®], IWE[®], IWORX[®], MUSAC[®], MuSLIC[®], OCTAT[®], OptiPort[®], POTSWIRE[®], QUAT[®], QuadFALC[®], SCOUT[®], SICAT[®], SICOFI[®], SIDEC[®], SLICOFI[®], SMINT[®], SOCRATES[®], VINETIC[®], 10BaseV[®], 10BaseVX[®] are registered trademarks of Infineon Technologies AG. 10BaseS[™], EasyPort[™], VDSLite[™] are trademarks of Infineon Technologies AG. Microsoft[®] is a registered trademark of Microsoft Corporation, Linux[®] of Linus Torvalds, Visio[®] of Visio Corporation, and FrameMaker[®] of Adobe Systems Incorporated.

CONFIDENTIAL
Table of Contents

	Preface	8
1	Introduction	9
2	Software Concept	10
2.1	T.38 FAX Relay Architecture	10
2.2	Memory Library	11
2.3	Timer Library	11
2.4	Signaling User Agent	12
2.5	Configuration Management	12
2.6	Resource Management	12
3	T.38 FAX Agent	13
3.1	Initialization	13
3.2	External Interface	14
3.3	FAX Data Packet Handling	15
3.4	T.38 FAX Agent Message Handling	15
3.5	Default Initialization Parameters	18
4	T.38 FAX Call Flow	19
5	Signaling User Agent Interface	21
5.1	Function Flow	21
5.2	Interface Description	22
5.2.1	Messages from the Signaling User Agent	22
5.2.1.1	IFIN_UA_START_T38_SESSION_REQ	22
5.2.1.2	IFIN_UA_END_T38_SESSION_REQ	22
5.2.1.3	IFIN_UA_SHUTDOWN_FA_REQ	23
5.2.2	Messages from the T.38 FAX Agent	23
5.2.2.1	IFIN_FA_END_FAX_CALL_REQ	23
5.2.2.2	IFIN_FA_START_T38_SESSION_RSP	23
5.2.2.3	IFIN_FA_START_T38_SESSION_ERR_RSP	23
5.2.2.4	IFIN_FA_END_T38_SESSION_RSP	23
5.2.2.5	IFIN_FA_END_T38_SESSION_ERR_RSP	23
5.2.3	Message Definition	23
5.2.3.1	x_IFIN_FA_Msg	23
5.2.3.2	x_IFIN_FA_StartT38SessReq	24
5.2.3.3	x_IFIN_FA_StartT38SessRsp	25
5.2.3.4	x_IFIN_FA_ErrorMsg	25
5.2.4	Enumerations	26
5.2.4.1	e_MT_T38_CapType	26
5.2.4.2	e_IFIN_FA_Error	26
5.2.4.3	e_IFIN_FA_EndT38Mode	27
6	Configuration Management Interface	28
6.1	Configuration Management module Interface	30
6.1.1	Messages from the Configuration Management	30
6.1.1.1	IFX_CM_FA_T38	30
6.1.2	Messages from the T.38 FAX-Agent	30
6.1.3	Message Definition	30
6.1.3.1	x_IFX_CM_Msg	30
6.1.3.2	x_IFX_CM_FA_T38Cfg	31

CONFIDENTIAL

7	Resource Management Interface	33
7.1	Resource Management Interface	34
7.1.1	Messages from the Resource Management	34
7.1.1.1	IFX_RM_RES_ALLOC_CODER	34
7.1.1.2	IFX_RM_RES_ALLOC_CODER_ERR	34
7.1.1.3	IFX_RM_RES_DEALLOC_CODER	34
7.1.1.4	IFX_RM_RES_DEALLOC_CODER_ERR	34
7.1.2	Messages from the T.38 FAX-Agent	35
7.1.3	Message Definition	35
7.1.3.1	x_IFX_RM_TxMsg	35
7.1.3.2	ux_IFX_RM_TxMsg	35
8	Platform Interface Module (PLT)	37
8.1	PLT Function Description	37
8.1.1	IFIN_FA_PLT_AllocateChannel	37
8.1.2	IFIN_FA_PLT_DeallocateChannel	37
8.1.3	IFIN_FA_PLT_DeviceRead	38
8.1.4	IFIN_FA_PLT_DeviceWrite	38
8.1.5	IFIN_FA_PLT_StartDemodulator	39
8.1.6	IFIN_FA_PLT_StartModulator	40
8.1.7	IFIN_FA_PLT_DisableDataPump	41
9	Source Code Organization	42
9.1	T.38 FAX Agent Source Code Organization	42
9.2	T.38 Protocol Stack Source Code Organization	42
	References	47

CONFIDENTIAL**List of Figures**

Figure 1	T.38 FAX Agent Data and Signal Flow Overview	9
Figure 2	Software Architecture of T.38 FAX Relay	11
Figure 3	Data Packet Handling	15
Figure 4	T.38 FAX-Agent Main Flow	17
Figure 5	T.38 FAX Agent Signaling Flow	19
Figure 6	Signaling User Agent Interface Handling	21
Figure 7	Configuration Management Interface Handling	29
Figure 8	Resource Management Interface Handling	33
Figure 9	Source Code Organization of PLT and T.38 FAX Agent	42
Figure 10	Source Code Organization of HDLC	43
Figure 11	Source Code Organization of OSIF	43
Figure 12	Source Code Organization of RTL	44
Figure 13	Source Code Organization of T.38	45
Figure 14	Source Code Organization of T38WRP	46

CONFIDENTIAL**List of Tables**

Table 1	Abbreviations	8
Table 2	FAX Channel Memory Requirement.	11
Table 3	Modulation, Demodulation Buffer Configuration	14
Table 4	Gain1, Gain2 Configuration	14
Table 5	Default Parameter	18
Table 6	Enumerations	26

Preface

Infineon Technologies offers FAX services over IP within the T.38 FAX Relay Package together with some VINETIC chipsets. The package includes the T.38 Protocol Stack, the T.38 FAX Agent and the documentation. Additional Test Software within the T.38 FAX Relay Package enables the user to use the T.38 Protocol Stack on the VINETIC evaluation package Easy 334 for development and demonstration purposes.

Documentation

The following documentation is available with the T.38 FAX Relay Package:

- T.38 Protocol Stack User's Manual Programmer's Reference
- T.38 Fax Agent User's Manual Programmer's Reference
- T.38 Test Application User's Manual Programmer's Reference
- VINETIC® T.38 FAX Relay Package Release Notes

Scope of the document

This document describes the system integration of the T.38 FAX Relay Package. For the system integration of the T.38 Protocol Stack an interface module is used which handles the signal flow between the T.38 Protocol stack, the system modules and the VINETIC. It also handles the data flow between the network and the VINETIC via the T.38 Protocol Stack. The interface module is called T.38 FAX Agent and is described within this documentation.

Abbreviations

Table 1 Abbreviations

Abbreviation	Full Form
RM	Resource Manager
CM	Configuration Module
Signaling UA	Signaling User Agent
TAPI	Telephony Application Interface
G3FE	Group 3 Facsimile Equipment

1 Introduction

With T.38 the ITU-T provides a standard based protocol for FAX Relay which specifies the transmission of the Internet Facsimile Protocol (IFP) data packets over the IP Network.

For integrating the T.38 Protocol Stack, additional software to the T.38 Protocol Stack is required which interfaces the T.38 Protocol interface to the Network Socket, the VINETIC Telephony Application (TAPI) and to the system applications (Resource Manager, Configuration Module, Signaling User Agent). The software is called "T.38 FAX Agent" and is part of the T.38 FAX Relay Package. The T.38 FAX Agent provides an interface to the system applications to control the T.38 FAX Relay software.

Figure 1 shows the signal and data flow for the T.38 FAX-Agent.

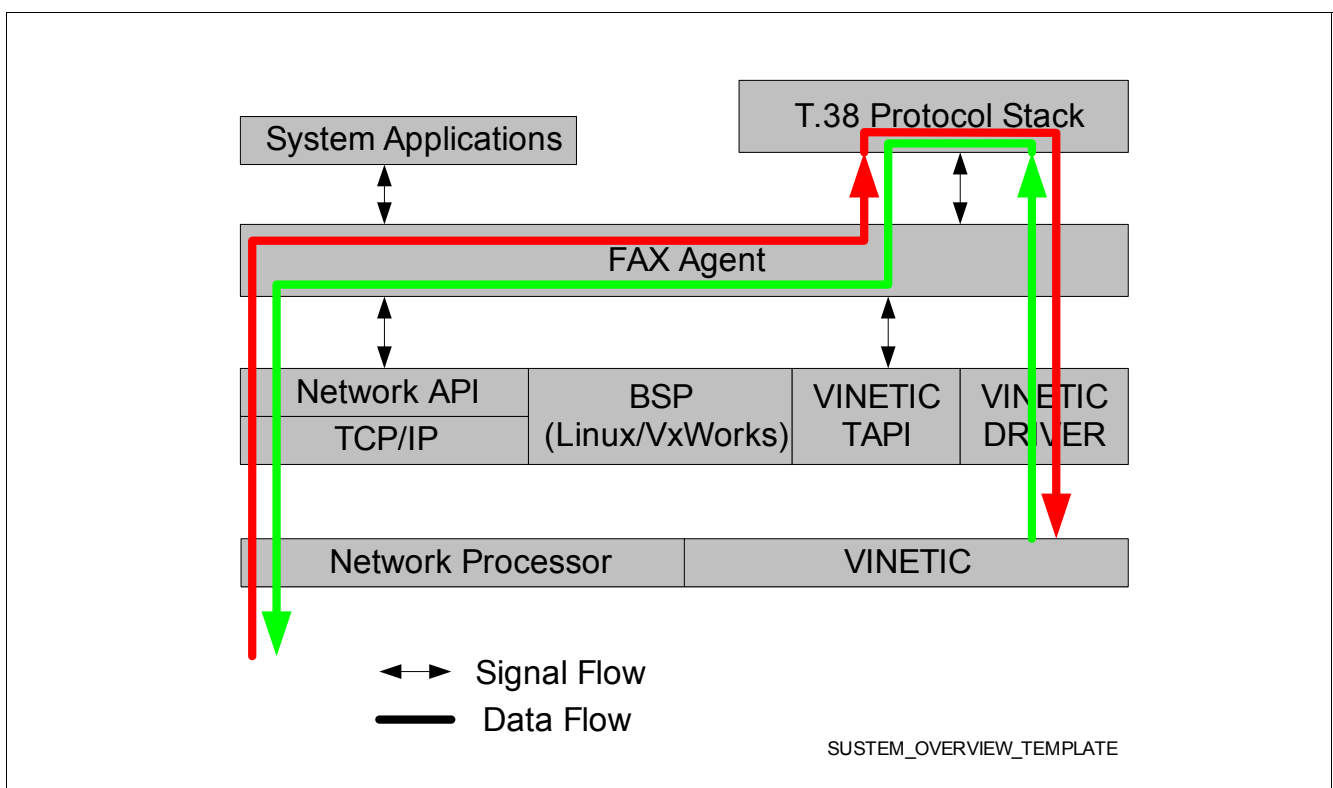


Figure 1 T.38 FAX Agent Data and Signal Flow Overview

2 Software Concept

The following chapter describes the software architecture for the T.38 FAX Relay concept. The T.38 FAX Relay Package consists of the T.38 Protocol Stack, the T.38 FAX Agent and the FAX Data Pump. The T.38 Protocol Stack is implemented on the network processor and the FAX Data Pump is running on the VINETIC.

The T.38 Protocol Stack is an OS independent implementation and the OS system dependencies are handled in the T.38 FAX Agent.

Note: The document describes the T.38 Protocol Stack implementation under Linux for the T.38 FAX Relay Package V1.1.

2.1 T.38 FAX Relay Architecture

This section describes the software modules used by the T.38 FAX Relay implementation. The T.38 FAX Agent is the main interface for the T.38 Protocol stack to the VINETIC and to the system applications. Between the T.38 Protocol Stack and the T.38 FAX Agent is a function based interface. The external T.38 FAX Agent interface to the system applications is a message-based interface to prevent blocking situations. The FAX Data Packets between the controller and the VINETIC are exchanged via the non blocking Driver Read/Write interface. The FAX Data Pump is controlled by the T.38 IO-Controls provided by the TAPI Driver Interface. The Read/Write and the IO-Control functions are implemented in the Platform Interface Module (PLT).

Interfaces handled by the T.38 FAX-Agent

The T.38 FAX Agent controls the data and message flow between the different modules and handles the following interfaces

- T.38 Protocol Stack Interface
- Network API
- TAPI Driver Interface
- Driver Data Read/Write Interface
- External Interface with message queues to Resource Manager (RM), Configuration Module (CM) and Signaling User Agent.

Figure 2 shows the software architecture for the T.38 FAX Relay implementation.

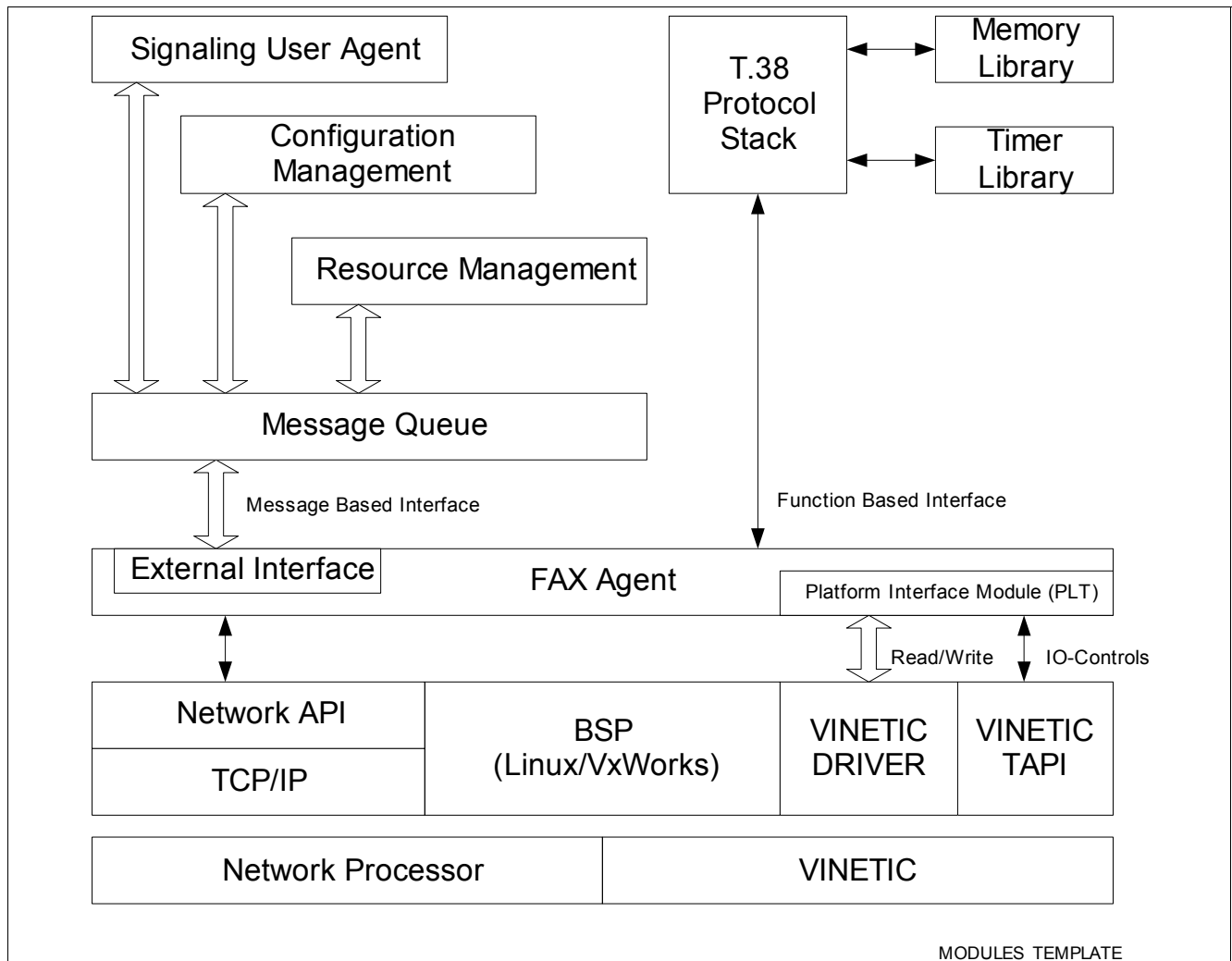


Figure 2 Software Architecture of T.38 FAX Relay

2.2 Memory Library

The Memory Library is used by T.38 Protocol Stack and is not part of the delivery. For each FAX channel 15.5 KByte of memory are required. If the trace facility is used the T.38 Protocol Stack allocates a circular buffer of additional 2 MB. The function interface for the Memory library is described in the “T.38 Protocol Stack Release” document. The system integrator has to provide the required memory functionality for the T.38 Protocol Stack.

Table 2 FAX Channel Memory Requirement

Buffer / Channel	Size [Bytes]
Data Pump Packet	514
Network Packet	2412
T.38 Channel	12764

2.3 Timer Library

The Timer Library is used by the T.38 Protocol Stack and is not part of the delivery. The function interface used for the Timer Library is described in the “T.38 Protocol Stack Release” document. The system integrator has to provide one 100 ms timer for each FAX channel.

2.4 Signaling User Agent

The Signaling User Agent handles the signaling protocol stack interface for SIP, H323 or MGCP/MEGACO. For example if the signaling protocol is SIP then the ITU-T T.38 Annex D call procedures are used for setting up the FAX call. In case of SIP the call setup is initiated by sending the SIP INVITE message and the VINETIC DSP is initialized for FAX procedures by switching from voice to FAX mode. The DIS detector is waiting for the V.21 preamble flags and indicates the detection of the V.21 preamble to the network processor. After that the signaling layer sends SIP INVITE packet and the Signaling User Agent indicates the T.38 session setup to the T.38 FAX Agent.

The Signaling User Agent is not part of the T.38 FAX Relay Package. The message based interface to the Signaling User Agent is described under the chapter "Signaling User Agent Interface".

2.5 Configuration Management

The Configuration Management (CM) module is used for the configuration of the T.38 FAX Relay Package and the configuration parameters are exchanged over a message based interface between the T.38 FAX Agent and the Configuration Management module. If the system does not provide a Configuration Module the configuration parameters can be stored in the configuration file "t38.ini".

The Configuration Management module is not part of the T.38 FAX Relay Package. The message based interface to the Configuration Management module is described under the chapter "Configuration Management Interface".

2.6 Resource Management

The Resource Management (RM) module is used for allocating the internal VINETIC resources. If the T.38 FAX Agent wants to setup a T.38 session the resources for the T.38 session are allocated by the Resource Management module. The Resource Management module is the administrator of the system resources.

The Resource Management module is not part of the T.38 FAX Relay Package. The message based interface to the Configuration Management module is described under the chapter "Resource Management Interface".

3 T.38 FAX Agent

This chapter describes the T.38 FAX Agent, the initialization, the main loop and the function flow of the main loop.

3.1 Initialization

This sub chapter describes the initialization of the T.38 FAX Agent.

Default VINETIC Initialization

The system application (Signaling User Agent) has to setup the default initialization for the VINETIC. The Default initialization includes the Analog Line Module configuration, the Signaling Module configuration and the Coder Module configuration for a voice transmission. The signal detection functionality provided by the Signaling Module for the CNG, CED and V.21 Preamble detection is used by the Signaling User Agent for the FAX machine indication.

T.38 FAX Agent Initialization

The T.38 FAX Agent is started together with the T.38 Protocol Stack in a separate Thread. The Signaling User Agent is calling the IFIN_FA_Init() function to create a Thread with the IFIN_FA_Main() function. The IFIN_FA_Main() function is the main loop for the T.38 FAX Agent and controls the T.38 Protocol Stack. The following code shows the IFIN_FA_Init() function called by the Signaling User Agent.

```
IFIN_FA_Init(void)
{
    v_nFaxProcId = fork();
    if (v_nFaxProcId > 0)
    {
        /* ### */
    }
    else if (v_nFaxProcId == 0)
    {
        IFIN_FA_Main();
    }
    else
    {
        /* Error */
        return IFIN_FA_FAIL;
    }
    return IFIN_FA_SUCCESS;
}
```

FAX Data Pump Buffer Configuration

The T.38 FAX Agent is configuring and starting the FAX Data Pump running in the Coder Module. After end of the FAX call the channel is reconfigure for voice transmission by the VINETIC TAPI.

The FAX Data Pump has internal buffers for the modulation and demodulation data. Data buffering is necessary to provide a continuous T.30 data stream to the Group 3 Facsimile Equipment (G3FE) terminals.

The parameters MODSM and MOBRD should be selected in such a way to get an acceptable compromise between controller reaction time and delay. The parameters listed in the following table are recommended parameters for the FAX Data Pump buffers.

Table 3 Modulation, Demodulation Buffer Configuration

Parameter	Value
MOBSM	320 (200ms)
MOBRD	233 (146ms)
DMBSD	32 (20ms)

FAX Data Pump Gain Configuration

The desired output signal power of the FAX Data Pump is set to -10dBm required for the power level of half duplex transmissions in accordance to the ITU-T V.2 recommendation.

Table 4 Gain1, Gain2 Configuration

Parameter	Value
Gain1	96
Gain2	128
DBM	10 (-10dBm)

3.2 External Interface

The external interface of the T.38 FAX Agent is a message-based interface as per figure 2 to avoid blocking situations for the external modules.

The message-based interface is implemented by using the File Descriptor mechanism which is supported by Linux as well as by VxWorks. Each process creates for a open device a list entry with a dedicated ID. The ID is also called File Descriptor. The application uses the select() function for waiting for the messages on the external interface. The select() function mechanism is also supported by Linux as well as by VxWorks.

The communication between the Signaling User Agent, the Resource Management and the Configuration Management is based on messages. Waiting for a message from the external interface is done with the select() function in the T.38 FAX Agent main loop.

The FD_ISSET() function checks if the File Descriptor for the Configuration Management is set and reads the message if there is a message from the Configuration Management available.

Example

```
memcpy(&vxTmpReadFds, &vxReadFds, sizeof(fd_set));
memcpy(&vxTmpWriteFds, &vxWriteFds, sizeof(fd_set));
nRetVal = select(FD_SETSIZE, &vxTmpReadFds, &vxTmpWriteFds, NULL, NULL);
/*.....*/
if (FD_ISSET(vnCmFaFifoFd, &vxTmpReadFds))
{
    /* Read message from CM*/
    nBytes = read(vnCmFaFifoFd, (char *) &xCmMsg, sizeof(x_IFX_CM_Msg));
    if (nBytes < 0)
        /* Message Error Handling */
        /* Process Message from CM */
        IFIN_FA_ProcessCmMsg(&xCmMsg);
}
```

3.3 FAX Data Packet Handling

The T.38 Protocol Stack is a function based implementation and is running in the critical data path. The T.38 FAX Agent main loop is blocking on the Read/Write File Descriptors. If there is a packet from the FAX Data Pump available the driver indicates the availability and the File Descriptor for the packet queue is set. The function IFIN_FA_PLT_DeviceRead() reads the packet from the Driver Interface and calls the function MT_T38_DataRcvdFromDataPump() to process the packet. During the processing of the packet by the T.38 Protocol Stack the main flow is blocked until the return from the T.38 Protocol Stack.

If the FAX Data Pump modulator request more data, the T.38 FAX Agent sends the data request to the T.38 Protocol Stack. The T.38 Protocol Stack calls then the function() IFIN_FA_SendDataToDataPump() to write the FAX Data Pump modulator data to the Driver Interface.

The critical issue is the processing time of the T.38 Protocol Stack. The processing time depends on the implementation of the Timer and Memory Library. For example using the function malloc() for the Memory Library implementation is to avoid. The function malloc() is a time consuming function and this would lead to FAX Data Pump packet loss because the main loop is waiting for the return from the stack. During this time a packet overflow in the Driver FIFO occurs.

Figure 3 shows the FAX Data Pump packet handling in the T.38 FAX-Agent main loop.

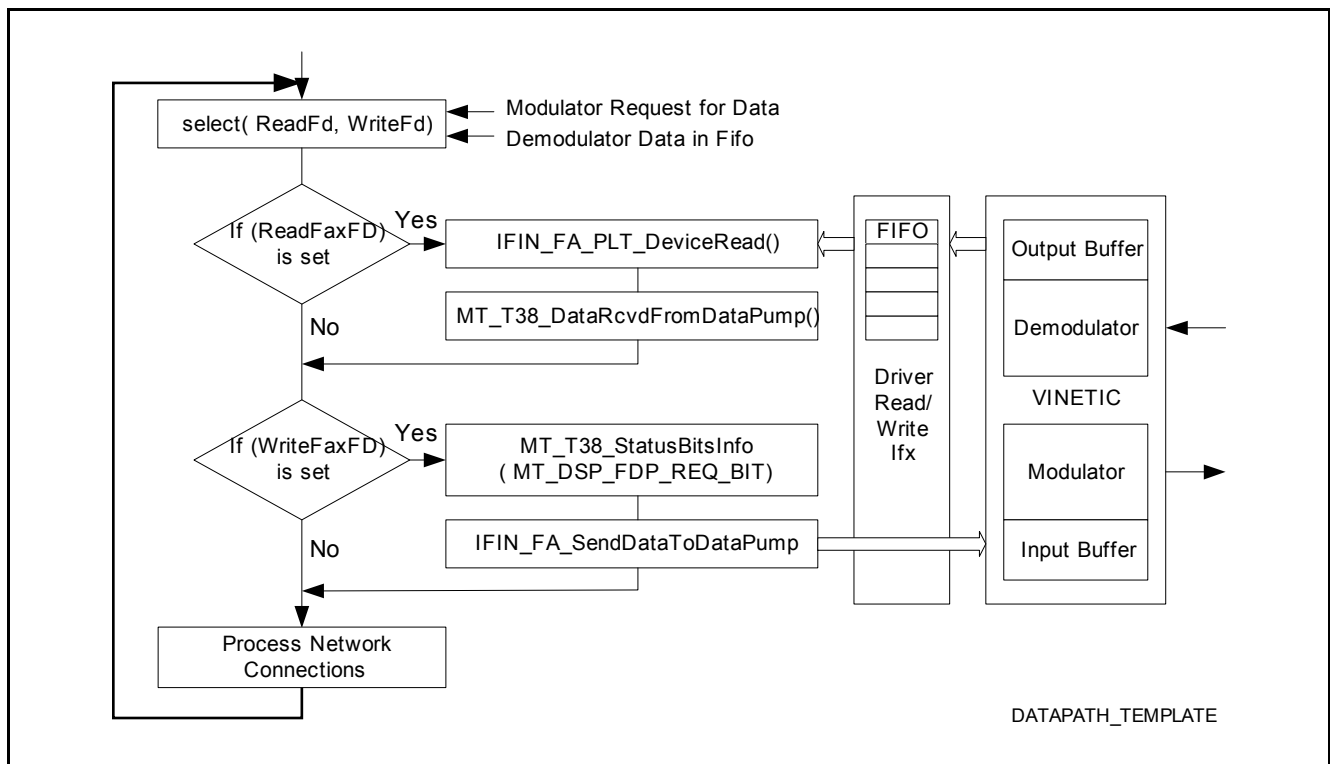


Figure 3 Data Packet Handling

3.4 T.38 FAX Agent Message Handling

This sub chapter explains the main loop flow of the T.38 FAX Agent. The main loop of the T.38 FAX Agent is processing the network connections and handles the message flow from the external interface. The main loop is implemented in the function IFIN_FA_MAIN().

There are three message pipes (FIFOs): one to the Configuration Management (CM), one to the Signaling User Agent (UA) and one to the Resource Management (RM) module.

The main loop checks if there is a message from the RM, the CM or the UA available. If there is a message from one of the external modules, the dedicated File Descriptor is set and the main loop calls the appropriate function to process the message.

The network connection has to be processed before starting checking the messages from the external interface. The reason is, if there is a request for setting up a new connection and a request to shut down a T.38 session, the request for the new connection has to be processed first.

For configuring the T.38 Protocol Stack the Configuration Management module sends a message including the configuration parameters to the T.38 FAX Agent. If the message is available the appropriate function `IFIN_FA_ProcessCmMsg()` is called for handling the configuration message.

For T.38 session setup, the T.38 FAX-Agent requests the VINETIC resources from the Resource Management module. If the resources are allocated the Resource Management sends a message with the necessary information about the resource to the T.38 FAX Agent. Then the T.38 FAX Agent calls the appropriate function `IFIN_FA_ProcessAllocCoderRspMsg()` for initializing the allocated VINETIC resources.

The Signaling User Agent sends messages to the T.38 FAX Agent to initialize the start of the T.38 session. The T.38 FAX Agent calls then the appropriate function `IFIN_FA_ProcessUaMsg()` to process the received message. During call setup and call shut down the T.38 FAX Agent informs the Signaling User Agent about the actual state.

Waiting for the message on the External interface by the `select()` function

```
memcpy(&vxTmpReadFds, &vxReadFds, sizeof(fd_set));  
memcpy(&vxTmpWriteFds, &vxWriteFds, sizeof(fd_set));  
nRetVal = select(FD_SETSIZE, &vxTmpReadFds, &vxTmpWriteFds, NULL, NULL);
```

Figure 4 gives an overview of the signal flow for the `IFIN_FA_MAIN()` function.

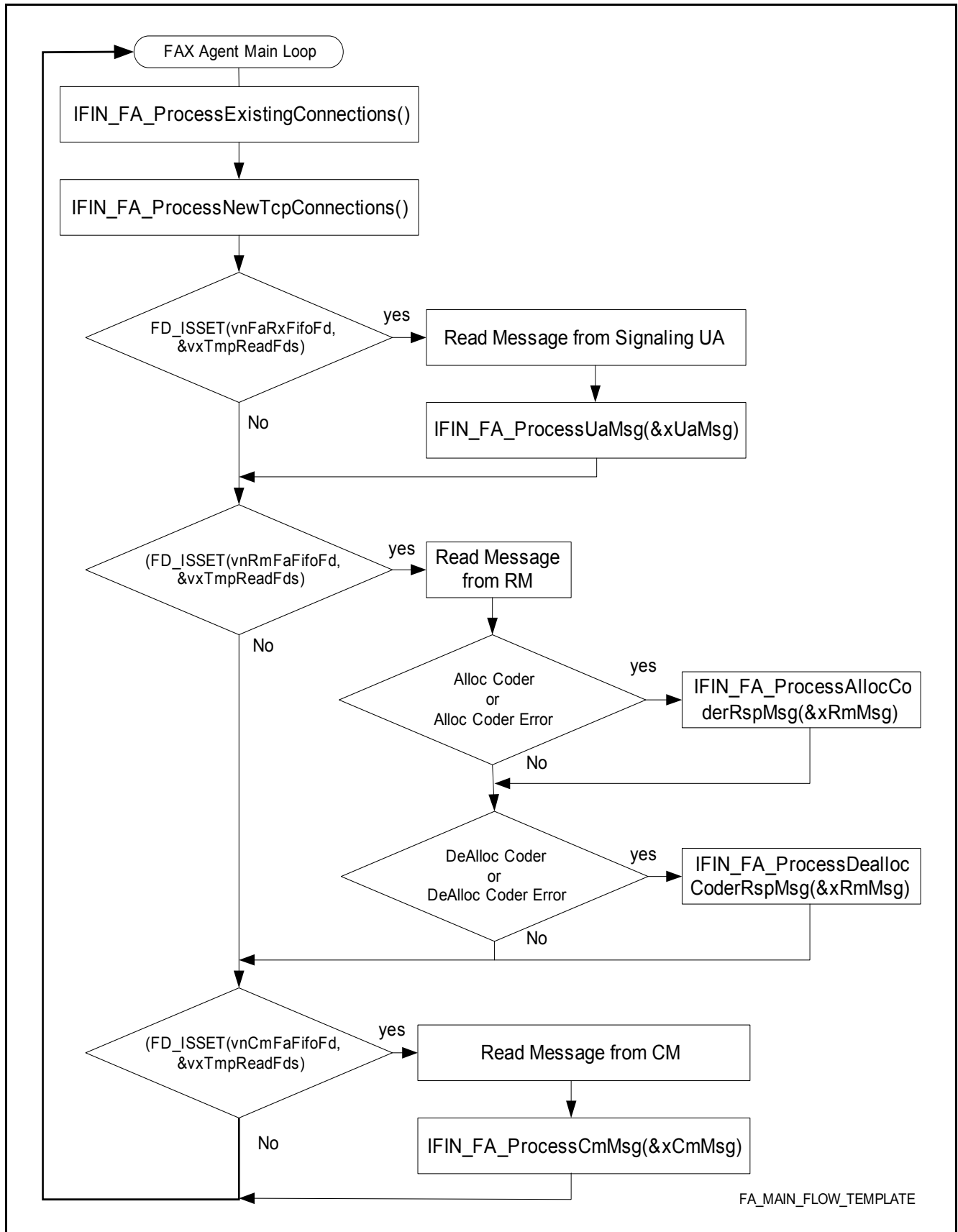


Figure 4 T.38 FAX-Agent Main Flow

3.5 Default Initialization Parameters

The following sub chapter describes the default configuration of the T.38 FAX Relay Package. The default initialization is done by function IFIN_FA_SetT38InitParam() before entering the main loop of the T.38 FAX Agent.

Default Parameter Setting

Table 5 Default Parameter

Parameter	Value
iOptions	0
uiDBmLevel	10
uiUdpHighRateErrRecoveryPackets	0
uiUdpLowRateErrRecoveryPackets	0
uiUdpPriorPacketsForFEC	0
uiNsxInfoFieldSz	0
uiDataWaitTime	0
unGain1	96
unGain2	128
ucDPNR	0
ucInputSignal	50
ucDataFrameLength	20
unMOBSM	320
unMOBRD	240
unDMBSD	32
uiAutoStartWaitTime	2000
uiAutoSpoofingTime	6000

4 T.38 FAX Call Flow

Figure 5 shows the signal flow between the T.38 FAX Agent and the external modules. The signal flow between the T.38 FAX Agent and the T.38 Protocol Stack is described in the “T.38 Protocol Stack Release” document.

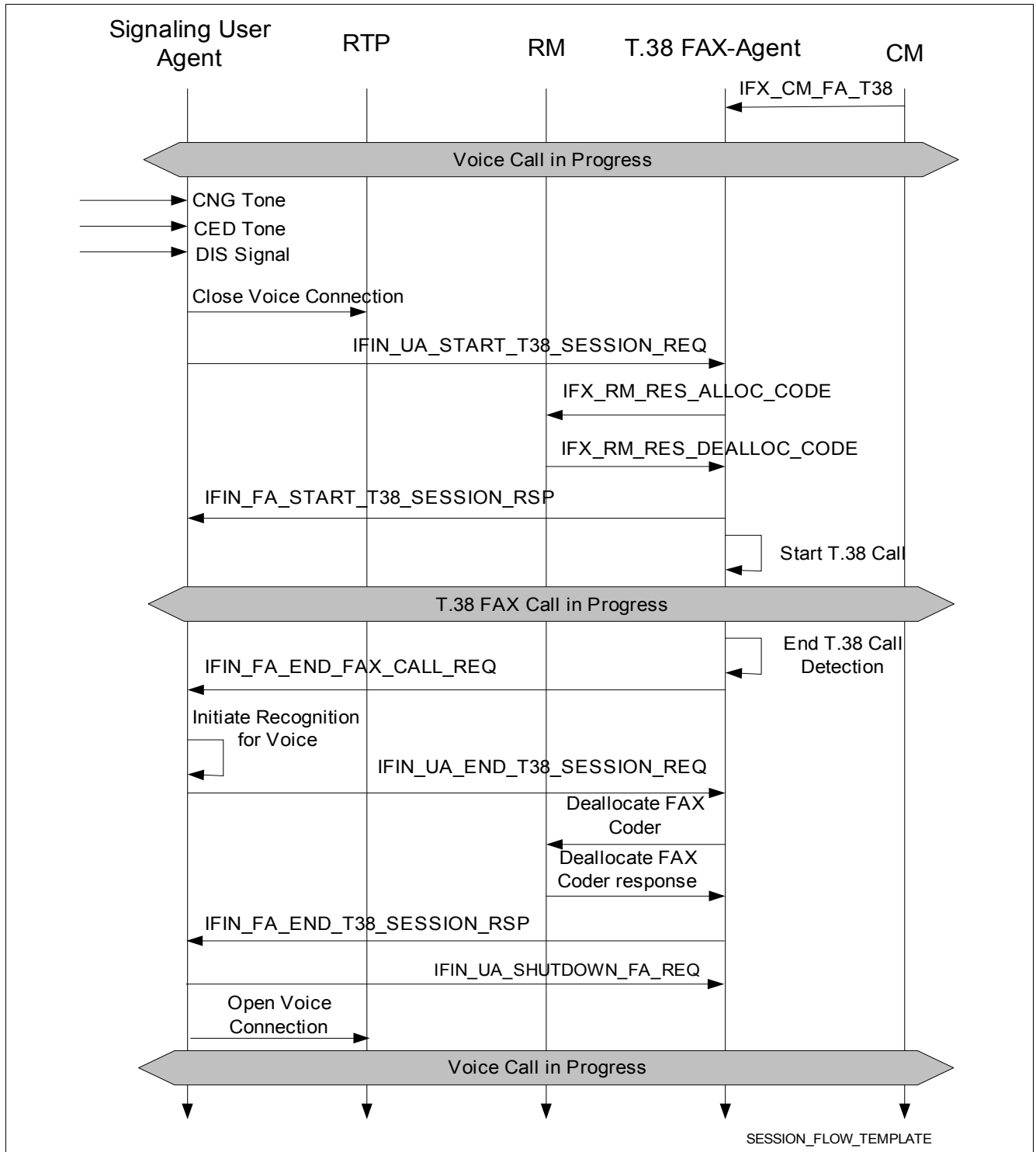


Figure 5 T.38 FAX Agent Signaling Flow

For setting up a FAX call, the VINETIC needs to be informed that a FAX connection is to be established. This works in the following: The VINETIC signal detection waits for the DIS (Digital Identification Signal) signal. After the DIS

signal is detected, the VINETIC informs the signaling layer (Signaling User Agent) about the FAX detection. Then the Signaling User Agent changes the media layer from Voice to FAX and informs the T.38 FAX Agent to setup the FAX session.

The T.38 FAX Agent requests the VINETIC resources from the Resource Manager. After successful allocation of the resources the T.38 session is started and controlled by the T.38 Protocol Stack.

If the FAX call is over the T.38 FAX Agent informs the Signaling User Agent about the end of the T.38 session. The Signaling User Agent sends then the indication to stop the T.38 session. The T.38 FAX Agent cleans up the network connection parameters, the connection table and informs the Signaling User Agent. After that the Signaling User Agent changes the media layer back from FAX to Voice.

The described scenario is an example for controlling the T.38 FAX Agent from the signaling layer. For using the SIP signaling protocol the ITU-T T.38 Annex D recommendation is used. For H323 it's the ITU-T T.38 Annex B recommendation.

5 Signaling User Agent Interface

The following chapter describes the external interface to the Signaling User Agent. The Signaling User Agent is an external module to the T.38 FAX Agent and handles the call signalling part of the signaling protocol (H323, SIP, MG). The message flow for the Signaling User Agent is as per figure 5.

5.1 Function Flow

The following sub chapter describes the message handling for the Signaling User Agent. If there is a message from the Signaling User Agent available, the IFIN_FA_ProcessUaMsg() function is called to process the message from the Call Setup Unit.

Figure 6 shows the signal flow for the function IFIN_FA_ProcessUaMsg().

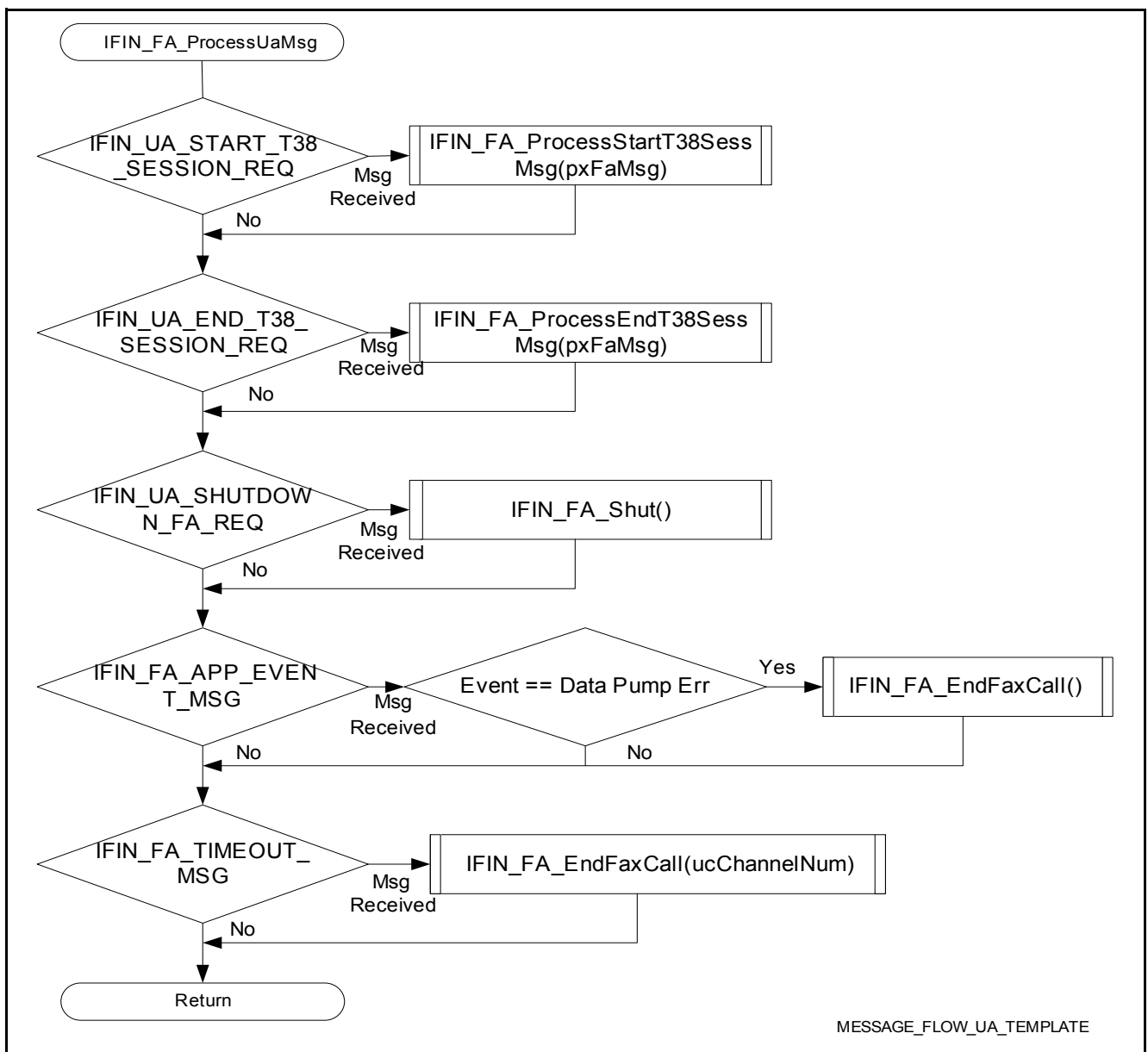


Figure 6 Signaling User Agent Interface Handling

The Signaling User Agent indicates with the message IFIN_UA_START_T38_SESSION REQ to start the T.38 session. The message IFIN_UA_END_T38_SESSION_REQ informs the T.38 FAX Agent to stop the T.38

session. If the Signaling User Agent needs to shut down the T.38 FAX Agent, the message IFIN_UA_SHUTDOWN_FA_REQ is send from the Signaling User Agent to the T.38 FAX Agent.

Receive Message from Signaling User Agent

```
if (FD_ISSET(vnFaRx FifoFd, &vxTmpReadFds))
{
    nBytes = read(vnFaRx FifoFd, (char *) &xUaMsg, sizeof(x_IFIN_FA_Msg));
    /* Check for Msg error */
    /* ... */
    /* Process Message */
    IFIN_FA_ProcessUaMsg(&xUaMsg);
}
```

Send Message to Signaling User Agent

```
/* Send IFIN_FA_END_FAX_CALL_REQ to UA */
memset(&xUaMsg, 0, sizeof(x_IFIN_FA_Msg));
xUaMsg.ucMsgType = IFIN_FA_END_FAX_CALL_REQ;
xUaMsg.ucChannelNum = ucChannelNum + 1;
if (IFIN_FA_SendMsgToUa(&xUaMsg) == IFIN_FA_FAIL)
{
    /* Error Handling */
}
```

5.2 Interface Description

The follwoing chapter describes the external message interface to the Signaling User Agent. The message is defined in the ifin_fa.h file.

5.2.1 Messages from the Signaling User Agent

The following sub chapter decribes the message types and their information sent from the Signaling User Agent to the T.38 FAX Agent. The messaging follows up the request response scheme. The response is only required from the T.38 FAX Agent and indicates the processing of a request message.

5.2.1.1 IFIN_UA_START_T38_SESSION_REQ

The Signaling User Agent sends the message type IFIN_UA_START_T38_SESSION_REQ to the T.38 FAX Agent to setup a T.38 session. The message is send by the Signaling User Agent after the negotiation of the T.38 parameters with the remote gateway. The T.38 session parameters are sent together with the message IFIN_UA_START_T38_SESSION_REQ to the T.38 FAX Agent. The message data is defined in the data structure x_IFIN_FA_StartT38SessReq.

5.2.1.2 IFIN_UA_END_T38_SESSION_REQ

The Signaling User Agent sends the message type IFIN_UA_END_T38_SESSION_REQ to the T.38 FAX Agent to terminate the T.38 session in case of:

- Received IFIN_FA_END_FAX_CALL_REQ from the T.38 FAX Agent
- Indication of the FAX termination from the remote gateway
- Local On-Hook detection

The message data is defined in the x_IFIN_FA_EndT38SessReq data structure.

5.2.1.3 IFIN_UA_SHUTDOWN_FA_REQ

The Signaling User Agent sends the message type IFIN_UA_SHUTDOWN_FA_REQ to the T.38 FAX Agent to shut down the T.38 FAX Agent.

5.2.2 Messages from the T.38 FAX Agent

The following sub chapter describes the message types and their information sent from the T.38 FAX Agent to the Signaling User Agent. The initialization and shut down of the T.38 session is signaled by the Signaling User Agent and the T.38 FAX Agent indicates events to the Signaling User Agent. The message flow is as per figure 5.

5.2.2.1 IFIN_FA_END_FAX_CALL_REQ

The T.38 FAX Agent sends the message type IFIN_FA_END_FAX_CALL_REQ to the Signaling User Agent in the case of End of FAX indication from the T.38 Protocol Stack. By sending the message type IFIN_FA_END_FAX_CALL_REQ the end of the T.38 session is indicated. The Signaling User Agent switches the media layer back from FAX to Voice.

5.2.2.2 IFIN_FA_START_T38_SESSION_RSP

The T.38 FAX Agent sends the message type IFIN_FA_START_T38_SESSION_RSP to the Signaling User Agent in case of successful T.38 session setup. The data for the IFIN_FA_START_T38_SESSION_RSP is defined in the x_IFIN_FA_StartT38SessRsp.

5.2.2.3 IFIN_FA_START_T38_SESSION_ERR_RSP

The message type IFIN_FA_START_T38_SESSION_ERR_RSP is send from the T.38 FAX Agent to the Signaling User Agent in case of T.38 session setup error.

5.2.2.4 IFIN_FA_END_T38_SESSION_RSP

The message type IFIN_FA_END_T38_SESSION_RSP is send from the T.38 FAX Agent to the Signaling User Agent in case of successfule T.38 session shut down.

5.2.2.5 IFIN_FA_END_T38_SESSION_ERR_RSP

The message type IFIN_FA_END_T38_SESSION_ERR_RSP is send from the T.38 FAX Agent to the Signaling user Agent in case of unsuccessful shut down of the T.38 session.

5.2.3 Message Definition

This sub chapter describes the definition of the Signaling User Agent interface message.

5.2.3.1 x_IFIN_FA_Msg

Description

The x_IFIN_FA_Msg describes the message used for the Signaling User Agent interface.

Prototype

```
typedef union{
    x_IFIN_FA_StartT38SessReq      xStartT38SessReq;
    x_IFIN_FA_StartT38SessRsp     xStartT38SessRsp;
    x_IFIN_FA_EndT38SessReq       xEndT38SessReq;
    x_IFIN_FA_ErrorMsg            xErrorMsg;
```

CONFIDENTIAL
Signaling User Agent Interface

```

} u_IFIN_FA_Msg;

typedef struct
{
    uchar8          ucMsgType;
    uchar8          ucChannelNum;
    uchar8          ucCallId;
    uchar8          ucSeqNo;
    u_IFIN_FA_Msg   uFaMsg;
} x_IFIN_FA_Msg;

```

Parameters

Data Type	Name	Description
x_IFIN_FA_StartT38SessReq	xStartT38SessReq	Message Data send from Signaling UA
x_IFIN_FA_StartT38SessRsp	xStartT38SessRsp	Message Data from T.38 FAX-Agent
x_IFIN_FA_EndT38SessReq	xEndT38SessReq	Message Data send from Signaling UA
uchar8	ucMsgType	Message Type <ul style="list-style-type: none"> • IFIN_FA_END_FAX_CALL_REQ • IFIN_FA_START_T38_SESSION_RSP • IFIN_FA_START_T38_SESSION_ERR_RSP • IFIN_FA_END_T38_SESSION_RSP • IFIN_FA_END_T38_SESSION_ERR_RSP • IFIN_UA_START_T38_SESSION_REQ • IFIN_UA_END_T38_SESSION_REQ • IFIN_UA_SHUTDOWN_FA_REQ
uchar8	ucChannelNum	Channel Number
uchar8	ucCallId	Message ID
uchar8	ucSeqNo	Sequence Number
u_IFIN_FA_Msg	uFaMsg	Message data content <ul style="list-style-type: none"> • x_IFIN_FA_StartT38SessReq • x_IFIN_FA_StartT38SessRsp • x_IFIN_FA_EndT38SessReq • x_IFIN_FA_AppEvent • x_IFIN_FA_ErrorMsg

5.2.3.2 x_IFIN_FA_StartT38SessReq
Description

The x_IFIN_FA_StartT38SessReq data structure describes the data used together with the message type IFIN_UA_START_T38_SESSION_REQ.

Prototype

```

typedef struct{
    uchar8          ucProtocol;
    uchar8          ucTcpTOC;
    char8           acRemoteIpAddress[IFIN_FA_MAX_IP_ADDR_LEN];
    uint16          unLocalPort;
}

```


CONFIDENTIAL
Signaling User Agent Interface

```

uint16          unRemotePort
x_IFIN_FA_Profile xProfile;
char8           cRtDelayFlag;
char8           cCallDir;
} x_IFIN_FA_StartT38SessReq;

```

Parameters

Data Type	Name	Description
uchar8	ucProtocol	Protocol type.
uchar8	ucTcpTOC	Type of Connection <ul style="list-style-type: none"> IFIN_FA_TCP_TOC_BIDIRECTIONAL IFIN_FA_TCP_TOC_UNIDIRECTIONAL
uchar8	acRemotelpAddr	IP Address of remote gateway
uint16	unLocalPort	Local TCP/UDP port number
uint16	unRemotePort	Local TCP/UDP port number
x_IFIN_FA_Profile	xProfile	Profile for Fax Agent
char8	cRtDelayFlag	Round trip delay flag. <ul style="list-style-type: none"> TRUE FALSE
char8	cCallDir	Calling Direction <ul style="list-style-type: none"> IFIN_FA_CALL_DIR_INCOMING IFIN_FA_CALL_DIR_OUTGOING

5.2.3.3 x_IFIN_FA_StartT38SessRsp
Description

The x_IFIN_FA_StartT38SessRsp data structure describes the data used together with the message type IFIN_FA_START_T38_SESSION_RSP.

Prototype

```

typedef struct{
    uint16          unLocalPort;
} x_IFIN_FA_StartT38SessRsp;

```

Parameters

Data Type	Name	Description
uint16	unLocalPort	Local TCP/UDP port number

5.2.3.4 x_IFIN_FA_ErrorMsg
Description

The x_IFIN_FA_ErrorMsg data structure describes the error message data send from the T.38 FAX Agent to the Signaling User Agent.

CONFIDENTIAL

Signaling User Agent Interface

Prototype

```
typedef struct{
    e_IFIN_FA_Error    eError;
} x_IFIN_FA_ErrorMsg;
```

Parameters

Data Type	Name	Description
e_IFIN_FA_Error	eError	T.38 FAX-Agent error code

5.2.4 Enumerations

The following section describes the enumerations used for the Signaling User Agent message interface.

Table 6 Enumerations

Name	Description
e_MT_T38_CapType	Description of T.38 Protocol stack capabilities
e_IFIN_FA_Error	Eoor Codes used by the T.38 FAX-Agent
e_IFIN_FA_EndT38Mode	Mode of T.38 termination

5.2.4.1 e_MT_T38_CapType

Description

The e_MT_T38_CapType enum describes the T.38 Protocol Stack capabilities.

Prototype

```
typedef enum{
    MT_T38_CapType_TCP,
    MT_T38_CapType_UDP,
} e_MT_T38_CapType;
```

Parameters

Name	Value	Description
MT_T38_CapType_TCP	0 _D	TCP
MT_T38_CapType_UDP	1 _D	UDP

5.2.4.2 e_IFIN_FA_Error

Description

The e_IFIN_FA_Error enum describes the Error Codes used by the T.38 FAX Agent.

Prototype

```
typedef enum{
    IFIN_FA_NO_ERROR = 0,
    IFIN_FA_GEN_ERROR,
    IFIN_FA_CHANNEL_NUMBER_ERROR,
```

CONFIDENTIAL

Signaling User Agent Interface

```

    IFIN_FA_FAX_PROFILE_ERROR,
    IFIN_FA SOCK_CREATE_ERROR,
    IFIN_FA_TCP_SERVER SOCK_CREATE_ERROR,
    IFIN_FA SOCK_SEND_ERROR,
    IFIN_FA_MAX_CONNS_EXCEED_ERROR,
    IFIN_FA_START_T38_SESSION_ERROR,
    IFIN_FA_RM_INTERFACE_ERROR,
    IFIN_FA_RESOURCE_UNAVAILABLE_ERROR,
    IFIN_FA_T38_SESSION_ERROR
} e_IFIN_FA_Error;

```

Parameters

Name	Value	Description
IFIN_FA_NO_ERROR	0 _D	No error
IFIN_FA_GEN_ERROR	1 _D	General error
IFIN_FA_CHANNEL_NUMBER_ERROR	2 _D	Wrong channel number
IFIN_FA_FAX_PROFILE_ERROR	3 _D	Error in profile for T.38 FAX-Agent.
IFIN_FA SOCK_CREATE_ERROR	4 _D	Socket creation failed
IFIN_FA_TCP_SERVER SOCK_CREATE_ERROR	5 _D	Socket creation in TCP Server failed
IFIN_FA SOCK_SEND_ERROR	6 _D	Sockets sending failed
IFIN_FA_MAX_CONNS_EXCEED_ERROR	7 _D	Error caused by exceeding maximum number of connections.
IFIN_FA_START_T38_SESSION_ERROR	8 _D	Start of T.38 session failed
IFIN_FA_RM_INTERFACE_ERROR	9 _D	Resource Manager Interface Error
IFIN_FA_RESOURCE_UNAVAILABLE_ERROR	10 _D	Unavailable resources
IFIN_FA_T38_SESSION_ERROR	11 _D	T.38 session error

5.2.4.3 e_IFIN_FA_EndT38Mode

Description

The e_IFIN_FA_ENDT38Mode enum describes the mode of T.38 termination.

Prototype

```

typedef enum{
    IFIN_FA_EndNormal = 0,
    IFIN_FA_EndOnHook
} e_IFIN_FA_EndT38Mode;

```

Parameters

Name	Value	Description
IFIN_FA_EndNormal	0 _D	Local termination or termination requested by remote gateway.
IFIN_FA_EndOnHook	1 _D	Local ON-Hook termination.

6 Configuration Management Interface

The Configuration Management is an external module to the T.38 FAX Agent and used for the configuration of the T.38 FAX Relay software.

The Configuration Module reads the configuration parameters for the T.38 protocol Stack from the "t38.ini" file. The Configuration Management module sends a message to the T.38 FAX Agent. After the message is received the T.38 FAX Agent calls the function IFIN_FA_ProcessCmMsg(). The function is setting the configuration parameters received from the Configuration Management. The parameters are set to the T.38 Protocol data structure x_MT_T38_InitParam defined by the T.38 Protocol Stack.

In case of a message error the message type IFX_CM_SET_ERR_RSP is sent to the Configuration Management module. If the configuration was successful the T.38 FAX Agent sends the message IFC_CM_SET_RSP back to the Configuration Management module.

The message flow for the Configuration Management is as per figure 5.

Figure 7 shows the handling of the message IFX_CM_FA_T38_CFG send by the Configuration Management module.

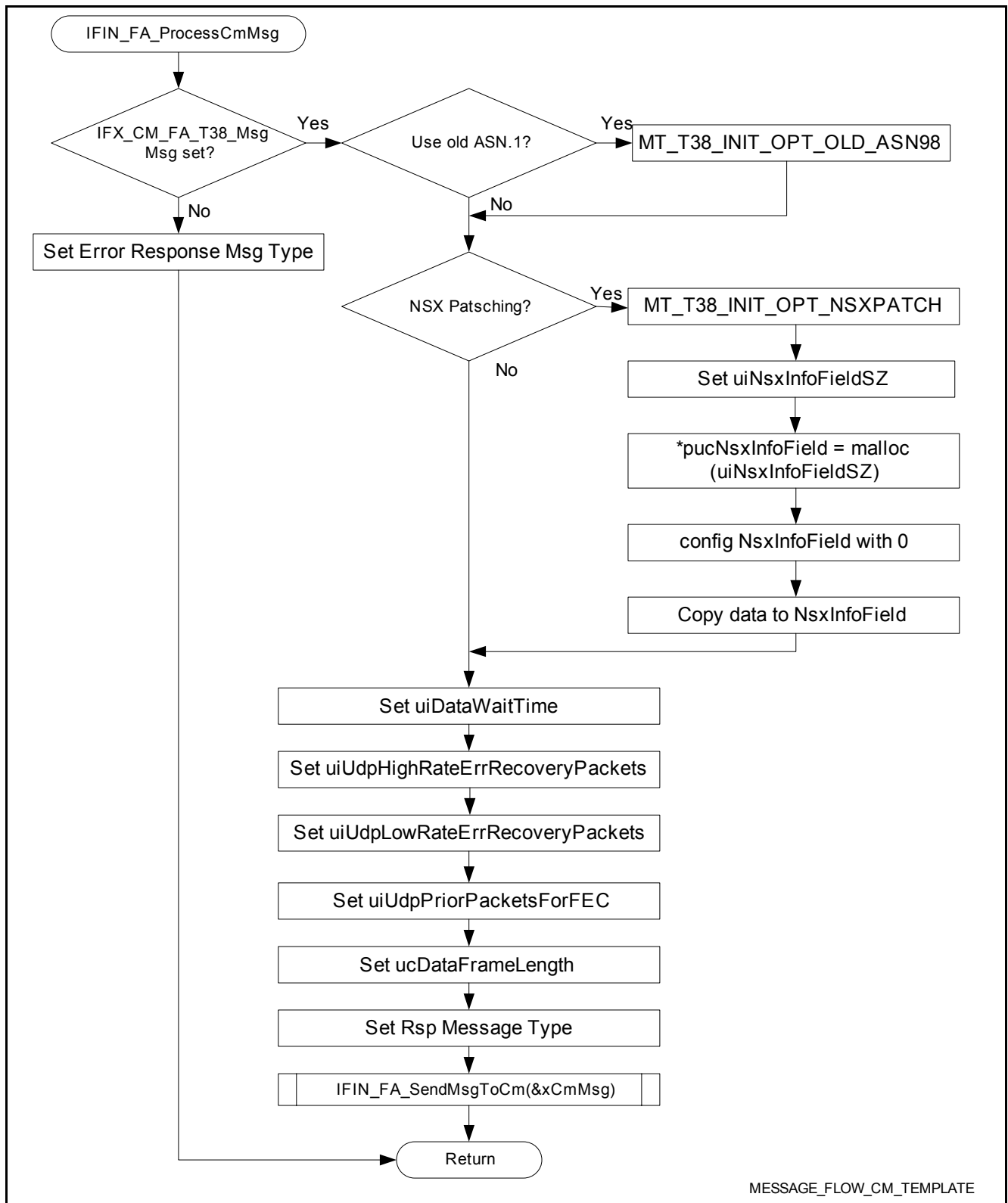


Figure 7 Configuration Management Interface Handling

Receive message from Configuration Management

```

if (FD_ISSET(vnCmFaFifoFd, &vxTmpReadFds))
{

```

```

/* Read and Process CM messages */
nBytes = read(vnCmFaFifoFd, (char *) &xCmMsg, sizeof(x_IFX_CM_Msg));
/* Message Error Handling*/
/*...*/
/* Process Message from CM */
IFIN_FA_ProcessCmMsg(&xCmMsg);
}

```

Send message to Configuration Management

```

/* Send Message IFX_CM_SET_RSP to CM */
memset(&xCmMsg, 0, sizeof(x_IFX_CM_Msg));
xCmMsg.ucChannel = pxCmMsg->ucChannel;
xCmMsg.ucInfoType = pxCmMsg->ucInfoType;
xCmMsg.ucMsgType = IFX_CM_SET_RSP
IFIN_FA_SendMsgToCm(&xCmMsg

```

6.1 Configuration Management module Interface

The following chapter describes the external message interface to the Configuration Management module. The message is defined in the ifx_cm.h file.

6.1.1 Messages from the Configuration Management

The following sub chapter describes the message types and their information sent from the Configuration Management module to the T.38 FAX Agent.

6.1.1.1 IFX_CM_FA_T38

The Configuration Management sends the IFX_CM_FA_T38 message info type to the T.38 FAX Agent to start the configuration of the T38 Protocol stack parameters.

6.1.2 Messages from the T.38 FAX-Agent

There are no message types defined from the T.38 FAX Agent to the Configuration Management module.

6.1.3 Message Definition

This sub chapter describes the definition of the Configuration Management module interface message.

6.1.3.1 x_IFX_CM_Msg

Description

The x_IFX_CM_Msg describes the message used for the Configuration Management interface.

Prototype

```

typedef struct
{
    uchar8 ucMsgType;
    uchar8 ucInfoType;
    uchar8 ucChannel;
    ux_IFX_CM_DataType uxData;
}

```

CONFIDENTIAL
Configuration Management Interface

```

} x_IFX_CM_Msg;

typedef union
{
    /* T38 Configuration */
    x_IFX_CM_FA_T38Cfg          xT38Cfg;
} ux_IFX_CM_DataType;

typedef enum
{
    IFX_CM_FA_T38_CFG,
} e_IFX_CM_InfoType;

typedef enum
{
    IFX_CM_GET_REQ=0,
    IFX_CM_SET_REQ,
    IFX_CM_GET_RSP,
    IFX_CM_GET_ERR_RSP,
    IFX_CM_SET_RSP,
    IFX_CM_SET_ERR_RSP
} e_IFX_CM_MsgType

```

Parameters

Data Type	Name	Description
uchar8	ucMsgType	Message Type
uchar8	ucInfoType	Message Info Type <ul style="list-style-type: none"> IFX_CM_FA_T38_CFG
uchar8	ucChannel	Channel
ux_IFX_CM_Msg	uxData	Message Type <ul style="list-style-type: none"> IFIN_FA_END_FAX_CALL_REQ

6.1.3.2 x_IFX_CM_FA_T38Cfg
Description

The x_IFX_CM_FA_T38Cfg data structure describes the data used together with the message info type IFX_CM_FA_T38_CFG. The parameters are part of the T.38 Protocol Stack and described more in detail in the "T.38 Protocol Stack Release" document. The parameters are used to configure the x_MT_T38_InitParam structure from the T.38 Protocol Stack. All other parameters are connection orientated and set during the start of the FAX Data Pump.

Prototype

```

typedef struct
{
    uchar8 ucOldAsn1;

```

CONFIDENTIAL
Configuration Management Interface

```

uchar8 ucNsxSize;
#define IFX_CM_FA_NSX_MAX_SIZE 40
uchar8 ucNsxInfoField[IFX_CM_FA_NSX_MAX_SIZE];
uchar8 ucDataWaitTime;
uint16 unUdpHRErrRecPkts;
uint16 unUdpLRErrRecPkts;
uint16 unUdpPriorFecPkts;
uint16 unFrameLength;
#define IFX_CM_FA_T38_CONN_UDP 1
#define IFX_CM_FA_T38_CONN_TCP 2
uchar8 ucT38Conn;
} x_IFX_CM_FA_T38Cfg;

```

Parameters

Data Type	Name	Description
uchar8	ucOldAsn1	ASN.1 Version
uchar8	ucNsxSize	Size of NSX patch
uchar8	ucNsxInfoField	Information Filed
uchar8	ucDataWaitTime	Data Wait Time
uint16	unUdpHRErrRecPkts	High Rate error recovery packets
uint16	unUdpLRErrRecPkts	Low Rate error recovery packets
uint16	unUdpPriorFecPkts	Prior Packets
uint16	unFrameLength	Frame Length
uchar8	ucT38Conn	T.38 Connection

7 Resource Management Interface

The Resource Management is an external module to the T.38 FAX Agent and handles the allocation of the VINETIC resources.

The Resource Management module sends a message with the information about the allocated VINETIC resource to the T.38 FAX Agent. After receiving the message the T.38 FAX Agent calls the function IFIN_FA_ProcessAllocCoderRspMsg() to initialize the allocated resource. The function is setting the FAX Data Pump parameters and is updating the network connection table.

After the T.38 session is finished the T.38 FAX Agent calls the function IFIN_FA_ProcessDeallocCoderRspMsg() to deallocate the VINETIC resource.

The message flow for the Resource Management is as per figure 5.

Figure 8 The following diagram shows the processing of the Resource Management message.

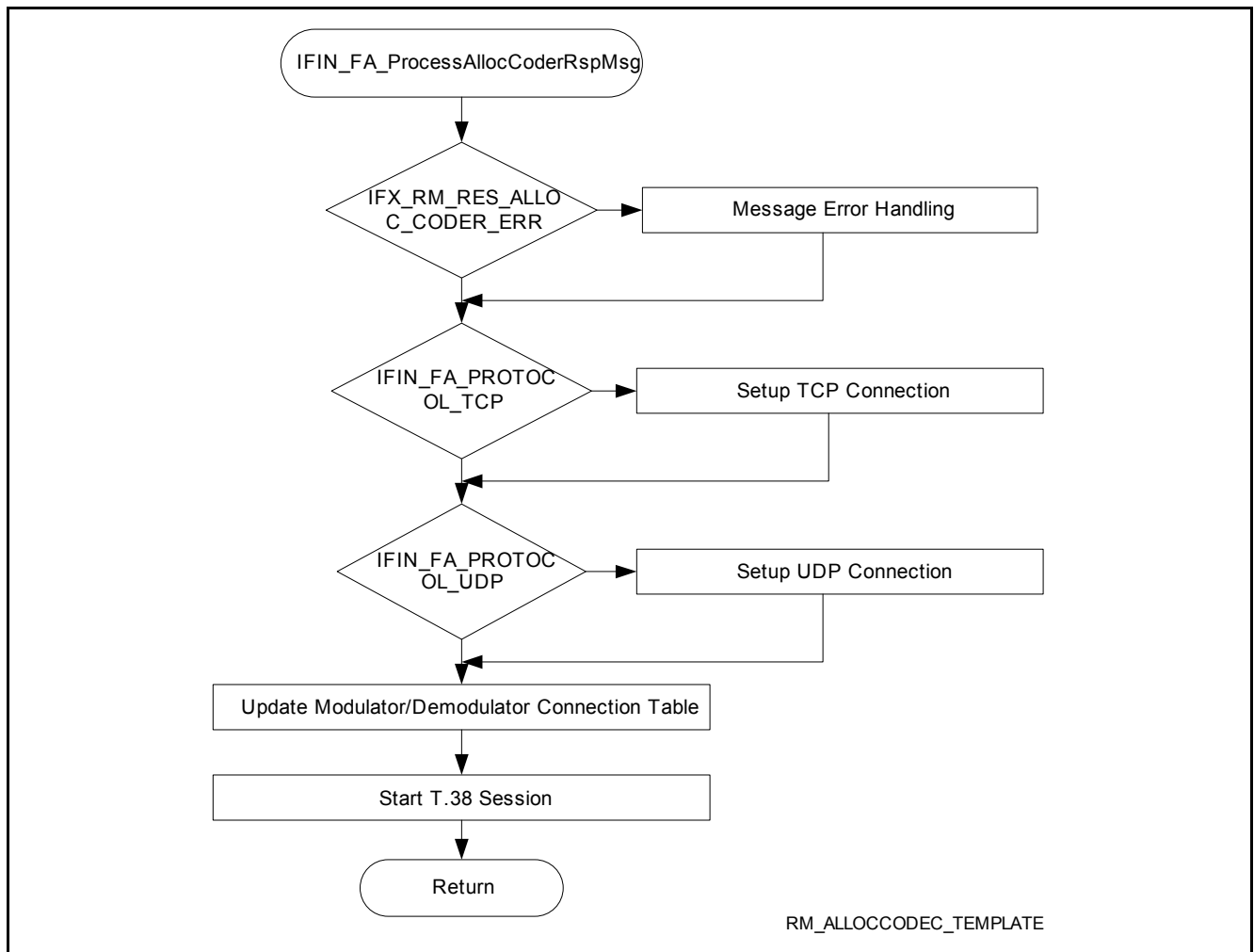


Figure 8 Resource Management Interface Handling

Receive message from Configuration Management

```

switch (xRmMsg.ucMsgType)
{
    case IFX_RM_RES_ALLOC_CODER:
    case IFX_RM_RES_ALLOC_CODER_ERR:

```

```

        IFIN_FA_ProcessAllocCoderRspMsg (&xRmMsg) ;
        break;
    case IFX_RM_RES_DEALLOC_CODER:
    case IFX_RM_RES_DEALLOC_CODER_ERR:
        IFIN_FA_ProcessDeallocCoderRspMsg (&xRmMsg) ;
        break;
    default:
        IFIN_FA_DBG (DBG_LVL_1,
            "FA - Unknown RM Msg %d\n", xRmMsg.ucMsgType) ;
        break;
}

```

Send message to Configuration Management

```

/* Write Message to RM Fifo */
if (IFIN_FA_SendMsgToRm (&xRmMsg) != IFIN_FA_SUCCESS)
{
    xFaMsg.ucMsgType = IFIN_FA_START_T38_SESSION_ERR_RSP;
    xFaMsg.uFaMsg.xErrorMsg.eError = IFIN_FA_RM_INTERFACE_ERROR;
    if (IFIN_FA_SendMsgToUa (&xFaMsg) != IFIN_FA_SUCCESS)
    {
        /* Error Handling */;
    }
}

```

7.1 Resource Management Interface

The following chapter describes the external message interface to the Resource Management module. The message is defined in the ifx_rm.h file.

7.1.1 Messages from the Resource Management

The following sub chapter describes the messages types and their information send from the Resource Management module to the T.38 FAX Agent.

7.1.1.1 IFX_RM_RES_ALLOC_CODER

The Configuration Management sends the IFX_RM_RES_ALLOC_CODER message type to the T.38 FAX Agent to configure the resources and to update the connection table in case of successful resource allocation.

7.1.1.2 IFX_RM_RES_ALLOC_CODER_ERR

The Configuration Management sends the IFX_RM_RES_ALLOC_CODER_ERR message type to the T.38 FAX Agent to update the connection table in case of unsuccessful resource allocation.

7.1.1.3 IFX_RM_RES_DEALLOC_CODER

The Configuration Management sends the IFX_RM_RES_DEALLOC_CODER message type to the T.38 FAX Agent to update the connection table in case of successful resource deallocation.

7.1.1.4 IFX_RM_RES_DEALLOC_CODER_ERR

The Configuration Management sends the IFX_RM_RES_DEALLOC_CODER_ERR message type to the T.38 FAX-Agent to update the connection table in case of unsuccessful resource deallocation.

7.1.2 Messages from the T.38 FAX-Agent

There are no message types defined from the T.38 FAX Agent to the Resource Management module.

7.1.3 Message Definition

This sub chapter describes the definition of the Resource Management module interface message.

7.1.3.1 x_IFX_RM_TxMsg

Description

The x_IFX_RM_TxMsg describes the message used for the Resource Management interface.

Prototype

```
typedef struct
{
    uchar8 ucMsgType;
    uchar8 ucAnaChnl;
    uchar8 ucCallId;
    ux_IFX_RM_TxMsg uxRmTxMsg;
} x_IFX_RM_TxMsg;
```

Parameters

Data Type	Name	Description
uchar8	ucMsgType	Message Type
uchar8	ucAnaChnl	Analog Channel
uchar8	ucCallId	Message ID
ux_IFX_RM_TxMsg	uxRmTxMsg	Message Data

7.1.3.2 ux_IFX_RM_TxMsg

Description

The ux_IFX_RM_TxMsg data structure contains the message data for the T.38 FAX Agent.

Prototype

```
typedef union
{
    x_IFX_RM_AllocAnalogRspMsg    xAllocAnalogRspMsg;
} ux_IFX_RM_TxMsg;

typedef struct
{
    uchar8 ucXAnaChnl;
    uchar8 ucXCallId;
} x_IFX_RM_AllocAnalogRspMsg;
```

CONFIDENTIAL

Resource Management Interface

Parameters

Data Type	Name	Description
uchar8	ucXAnaChnl	Analog Channel
uchar8	ucXCallID	Call ID

8 Platform Interface Module (PLT)

The VINETIC Driver Read/Write interface is used to exchange data packets between the T.38 FAX Agent and the VINETIC.

The FAX T.38 Services provided by the TAPI are implemented in the Platform Interface Module functions which are called by the T.38 FAX Agent.

The Function Platform Interface gives the flexibility to adapt the T.38 FAX Agent to different system architectures. The following IO-Controls are used by the T.38 FAX-Agent:

FAX TAPI Services

- IFXPHONE_FAX_SETMOD
- IFXPHONE_FAX_SETDEM
- IFXPHONE_FAX_DISABLE
- IFXPHONE_FAX_STATUS

8.1 PLT Function Description

This sub chapter describes the functions provided by the PLT interface.

8.1.1 IFIN_FA_PLT_AllocateChannel

Description

The function IFIN_FA_PLT_AllocateChannel() uses the IFXPHONE_DATA_ADD IO-Control from the TAPI interface. The parameters used for the configuration are defined in the x_IFIN_DataChannelParams data structure.

Prototype

```
IFIN_FA_PLT_AllocateChannel(ucChannelNum IN uchar8 ucChannelNum,
                           IN uchar8 ucCoder,
                           OUT int16 *pnFaxFd)
```

Parameters

Data Type	Name	Description
uchar8	ucChannelNum	Phone Channel Number
uchar8	ucCoder	Coder Number
int16	*pnFaxFd	Pointer to FAX File Descriptor

8.1.2 IFIN_FA_PLT_DeallocateChannel

Description

The function IFIN_FA_PLT_DeallocateChannel() is not used.

Prototype

```
IFIN_FA_PLT_DeallocateChannel(IN uchar8 ucChannelNum,
                              IN uchar8 ucCoder,
                              IN int16 nFaxFd)
```

CONFIDENTIAL

Platform Interface Module (PLT)

Parameters

Data Type	Name	Description
uchar8	ucChannelNum	Phone Channel Number
uchar8	ucCoder	Coder Number
int16	*pnFaxFd	Pointer to FAX File Descriptor

8.1.3 IFIN_FA_PLT_DeviceRead

Description

The IFIN_FA_PLT_DeviceRead() function is called by the IFIN_FA_ProcessExistingConnections() function. The function reads data from parallel interface with the Driver Read function.

The IFIN_FA_ProcessExistingConnections() function is checking the file descriptors with FD_ISSET() on each Phone Channel if the file descriptor is set. If the file descriptor is set the IFIN_FA_PLT_DeviceRead() function is called.

Prototype

```
IFIN_FA_PLT_DeviceRead(IN int32 nFaxFD, OUT char8 *pcBuf, IN uint16 unBufLen)
```

Parameters

Data Type	Name	Description
int32	nFaxFd	File descriptor
ichar8	*pcBuf	Pointer to Data Buffer
uint16	unBufLen	Buffer Length

8.1.4 IFIN_FA_PLT_DeviceWrite

Description

The IFIN_FA_PLT_DeviceWrite() function is used to write the FAX data to the Driver Write interface function.

Prototype

```
IFIN_FA_PLT_DeviceWrite(IN int32 nFaxFd,
                        IN char8 *pcFaxBuf,
                        IN uint16 unBufLen)
```

Parameters

Data Type	Name	Description
int32	nFaxFd	File Descriptor
ichar8	*pcFaxBuf	Pointer to FAX data buffer
uint16	unBufLen	Buffer Length

8.1.5 IFIN_FA_PLT_StartDemodulator

Description

The IFIN_FA_PLT_StartDemodulator() function is used to start the FAX Data Pump.

Prototype

```
IFIN_FA_PLT_StartDemodulator(IN int16 nFaxFd,
                             IN uchar8 ucTRN,
                             IN uchar8 ucEQ,
                             IN uchar8 ucSTD2,
                             IN uchar8 ucSTD1,
                             IN uchar8 ucDPNR,
                             IN uint16 unGain1,
                             IN uint16 unGain2,
                             IN uint16 unMOBSM,
                             IN uint16 unMOBRD,
                             IN uint16 unDMBSD)
```

Parameters

Data Type	Name	Description
int16	nFaxFd	File Descriptor
uchar8	ucTRN	Short Training Sequence <ul style="list-style-type: none"> TRN=0 - Short training sequence TRN=1 - Long training sequence
uchar8	ucEQ	<ul style="list-style-type: none"> EQ=0 - Reset equalizer before demodulation EQ=1 - Reuse of the previous used equalizer coefficients
uchar8	ucSTD2	Alternative desired standard <ul style="list-style-type: none"> STD2 = 0 0001 - V.21 STD2 = 0 0010 - V.27ter / 2400 STD2 = 0 0011 - V.27ter / 4800 SDT2 = 0 0100 - V.29 / 7200 STD2 = 0 0101 - V.29 / 9600 STD2 = 0 0110 - V.17 / 7200 STD2 = 0 0111 - V.17 / 9600 STD2 = 0 1001 - V.17 / 14400 STD2 = 1 1111 - don't used STD2, only STD1 is used
uchar8	ucSTD1	Alternative desired standard <ul style="list-style-type: none"> STD1 = 0 0001 - V.21 STD1 = 0 0010 - V.27ter / 2400 STD1 = 0 0011 - V.27ter / 4800 SDT1 = 0 0100 - V.29 / 7200 STD1 = 0 0101 - V.29 / 9600 STD1 = 0 0110 - V.17 / 7200 STD1 = 0 0111 - V.17 / 9600 STD1 = 0 1001 - V.17 / 14400
uchar8	ucDPNR	Data Pump Resource number
uint16	unGain1	Gain for upstream

Data Type	Name	Description
uint16	unGain2	Gain for downstream
uint16	unMOBSM	Modulation buffer, level for start modulation
uint16	unMOBRD	Modulation buffer, request more data
uint16	unDMBSD	Demodulation buffer, send data level

8.1.6 IFIN_FA_PLT_StartModulator

Description

The IFIN_FA_PLT_StartModulator() function is used to configure the FAX Data Pump modulator.

Prototype

```
IFIN_FA_PLT_StartModulator(IN int16 nFaxFd,
                           IN uchar8 ucDBM,
                           IN uchar8 ucTEP,
                           IN uchar8 ucTRN,
                           IN uchar8 ucSTD,
                           IN uint16 unSGLLEN,
                           IN uchar8 ucDPNR,
                           IN uint16 unGain1,
                           IN uint16 unGain2,
                           IN uint16 unMOBSM,
                           IN uint16 unMOBRD,
                           IN uint16 unDMBSD)
```

Parameters

Data Type	Name	Description
int16	nFaxFd	File Descriptor
uchar8	ucDBM	Desired output signal in -dBm
uchar8	ucTEP	<ul style="list-style-type: none"> TEP=0 - No TEP TEP=1 - Generate TEP
uchar8	ucTRN	<ul style="list-style-type: none"> TRN=0 - Short training sequence TRN=1 - Long training sequence
uchar8	ucSTD	<ul style="list-style-type: none"> STD = 0 0000 - silence STD = 0 0001 - V.21 STD = 0 0010 - V.27ter / 2400 STD = 0 0011 - V.27ter / 4800 STD = 0 0100 - V.29 / 7200 STD = 0 0101 - V.29 / 9600 STD = 0 0110 - V.17 / 7200 STD = 0 0111 - V.17 / 9600 STD = 0 1000 - V.17 / 12000 STD = 0 1001 - V.17 / 14400 STD = 0 1010 CNG STD = 0 1011 CED
uint16	unSGLLEN	Signal length

CONFIDENTIAL
Platform Interface Module (PLT)

Data Type	Name	Description
uchar8	ucDPNR	Data Pump Resource number
uint168	unGain1	Gain for upstream
uint16	unGain2	Gain for downstream
uint16	unMOBSM	Modulation buffer, level for start modulation
uint16	unMOBRD	Modulation buffer, request more data
uint16	unDMBSD	Demodulation buffer, send data level

8.1.7 IFIN_FA_PLT_DisableDataPump

Description

The IFIN_FA_PLT_DisableDataPump() function is used to disable the FAX Data Pump.

Prototype

```
IFIN_FA_PLT_DisableDataPump(IN uchar8 ucChannelNum,
                             IN int16 nFaxFd)
```

Parameters

Data Type	Name	Description
uchar8	ucChannelNum	Channel Number
int16	nFaxFd	File Descriptor

9 Source Code Organization

The following chapter gives an overview of the source code organization of the T.38 FAX Relay Package. How to build the package is described in the “T.38 Test Application Release” document.

9.1 T.38 FAX Agent Source Code Organization

Figure 9 shows the source code organization of the T.38 FAX Agent and the PLT module.

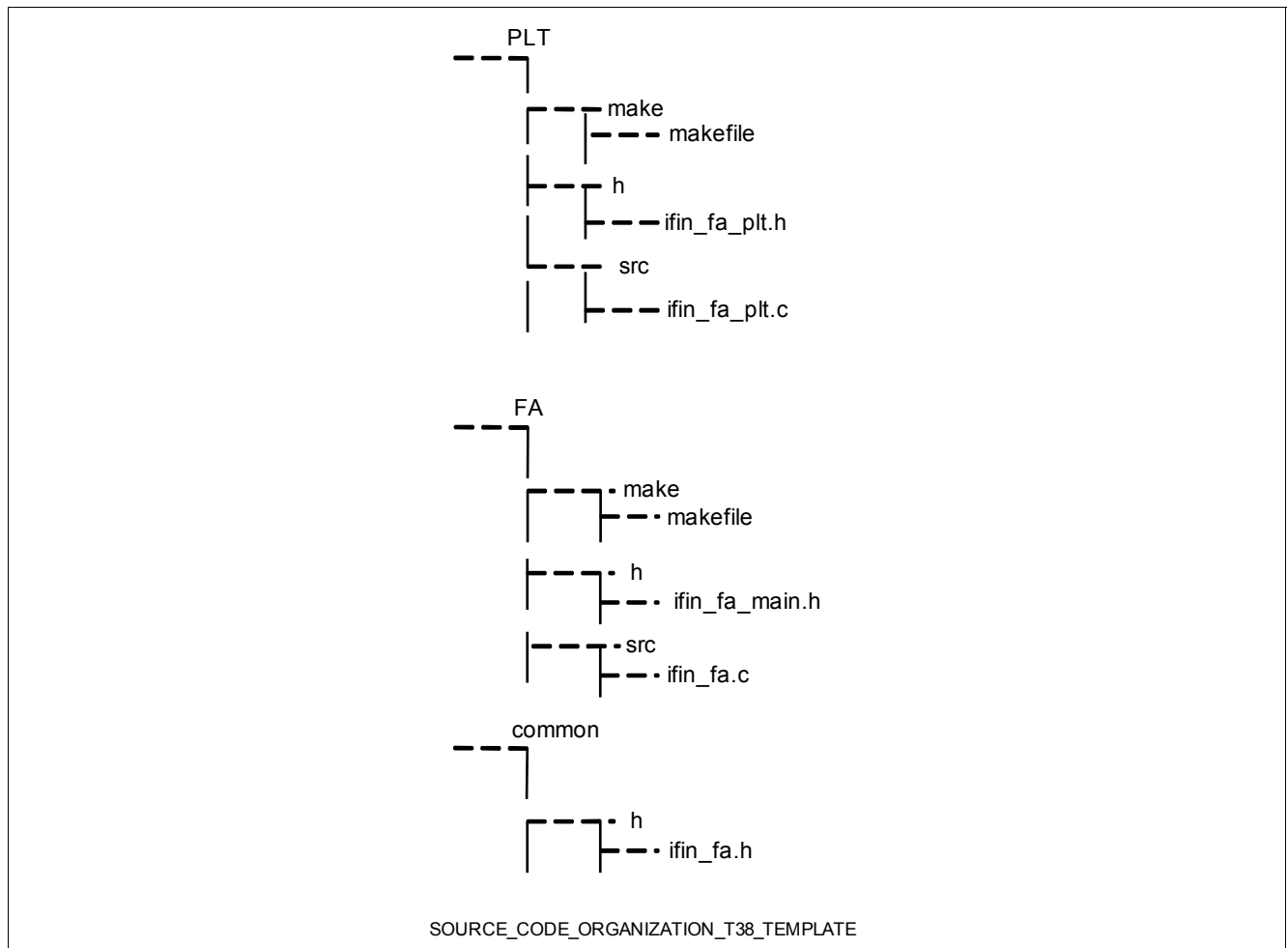


Figure 9 Source Code Organization of PLT and T.38 FAX Agent

9.2 T.38 Protocol Stack Source Code Organization

The following source code is part of the T.38 Protocol Stack. For detailed information refer to the “T.38 Protocol Stack Release” document.

Figure 10 shows the source code organization of the HDLC module.

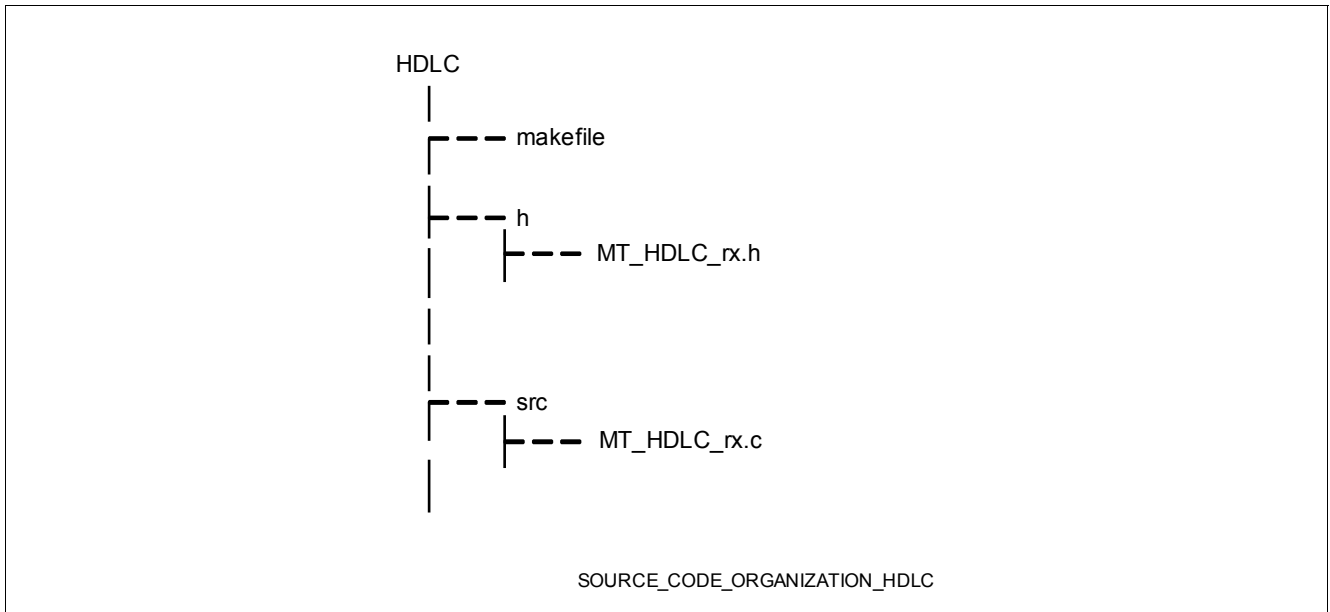


Figure 10 Source Code Organization of HDLC

Figure 11 shows the source code organization of the OSIF module.

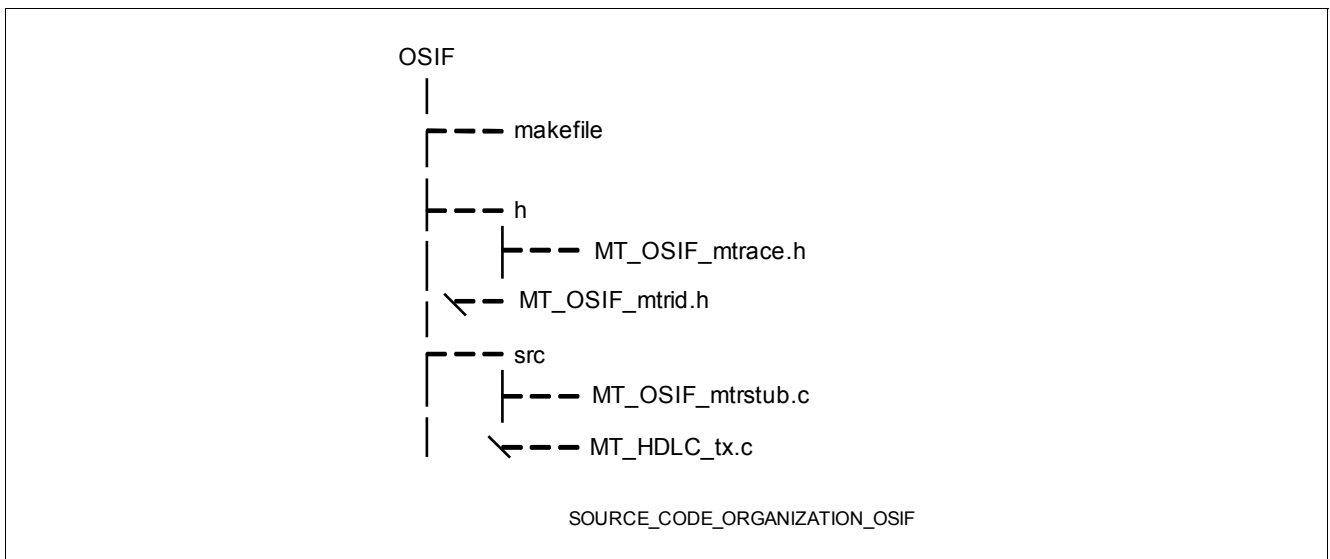


Figure 11 Source Code Organization of OSIF

Figure 12 shows the source code organization of the RTP module.

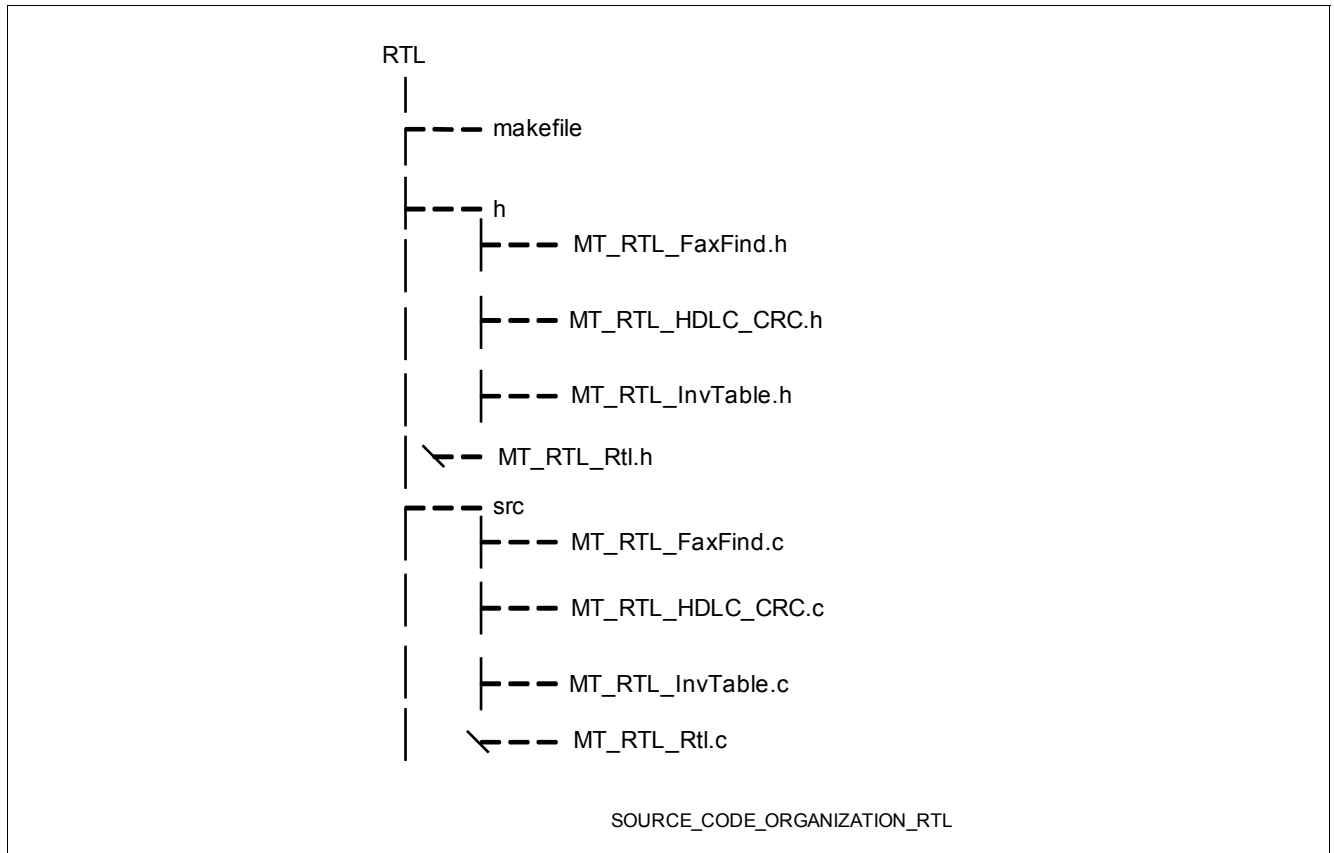


Figure 12 Source Code Organization of RTL

Figure 13 shows the source code organization of the T38 module.

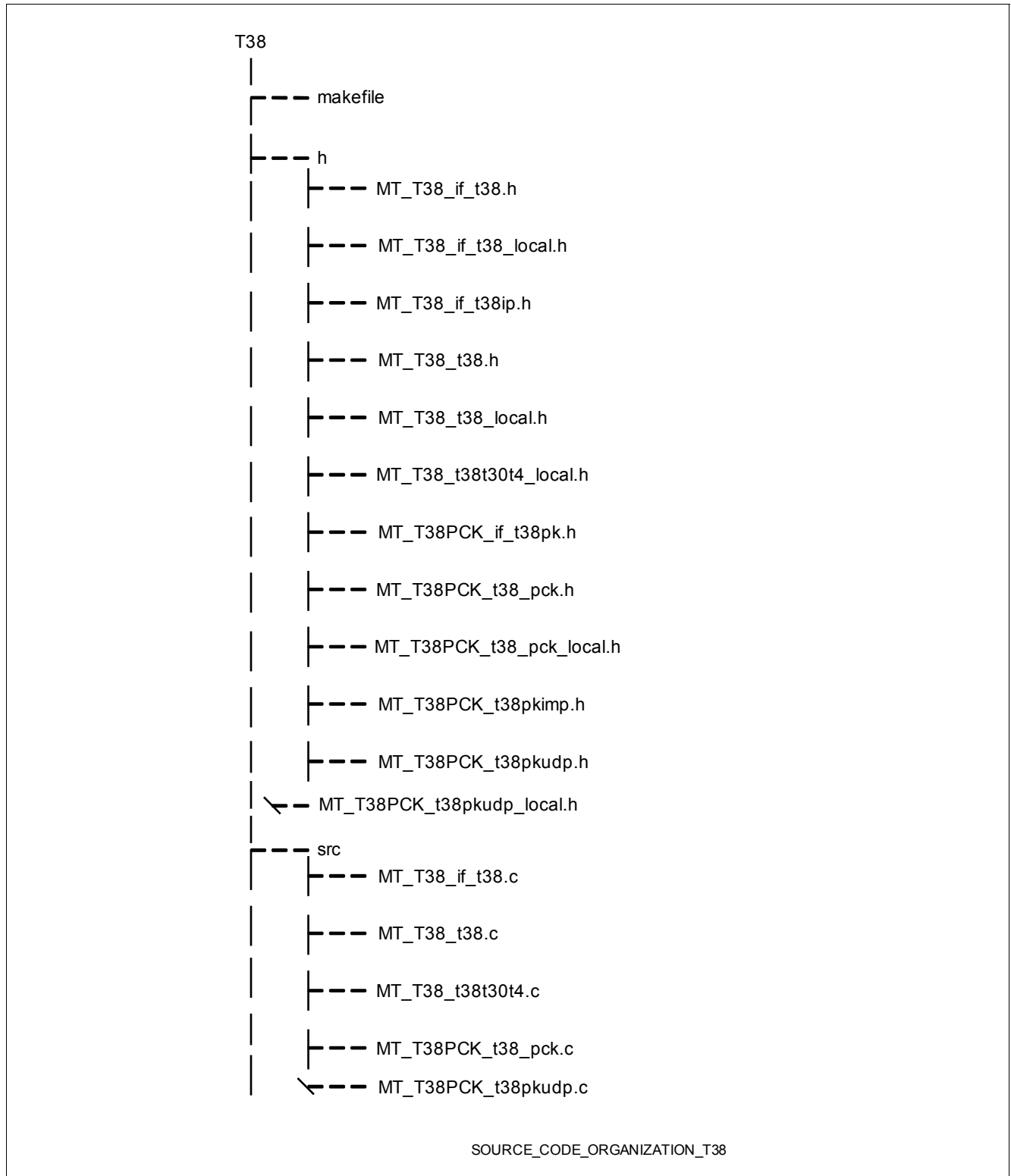


Figure 13 Source Code Organization of T.38

Figure 14 shows the source code organization of the T38WRP module.

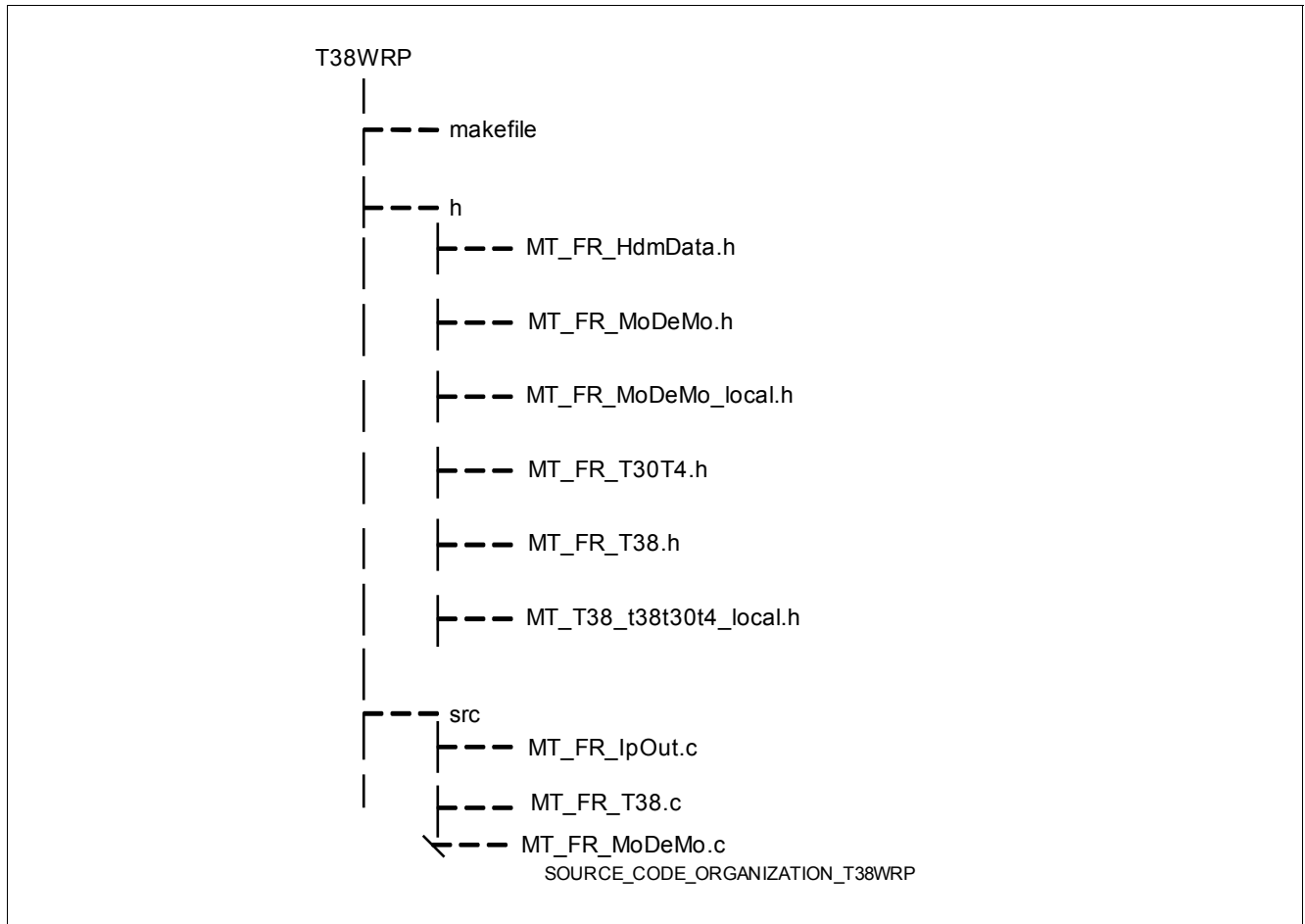


Figure 14 Source Code Organization of T38WRP

References

- [1] VINETIC® Version 1.4/2.1 Prel. User's Manual – EDSP Firmware Description Rev. 1.0, 2004-06-18
- [2] VINETIC® Version 1.4/2.1/2.2 Prel. User's Manual – EDSP Firmware Description Rev. 2.0, in preparation
- [3] T.38 Protocol Stack Release 1.16 User's Manual Programmer's Reference Rev. 1.0
- [4] T.38 Test Application Release 1.0 User's Manual Programmer's Reference Rev. 1.0
- [5] VINETIC® (PEB/PEF 33xy) Version 1.4 and 2.1/2.2 Application Note Telephony API (TAPI) V2.3 Rev. 16, 2005-04-13
- [6] VINETIC® Prel. User's Manual Software Description - Driver Rev. 4.0, 2004-12-08
- [7] VINETIC® Driver Software Release Notes
- [8] VINETIC® T.38 FAX Relay Package Release Notes

www.infineon.com