

VINETIC®

Voice and Internet Enhanced Telephony Interface
Circuit

T.38 Protocol Stack Release 1.16

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 Protocol Stack Release 1.16, Voice and Internet Enhanced Telephony Interface Circuit

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	T.38 Protocol Stack	10
2.1	Modules of T.38 Protocol Stack	11
2.1.1	HDLC Module	12
2.1.2	OSIF Module	13
2.1.3	RTL Module	14
2.1.4	T38 Module	14
2.1.5	T38WRP Module	14
2.2	Data Pump	14
3	T.38 Configuration Parameters	16
3.1	T.38 Session Parameters for Capability Negotiation	16
3.2	T.38 Protocol Stack Parameters	16
3.3	Data Pump Parameters	17
4	Interface Description	19
4.1	Basic Type Definitions	21
4.1.1	uint8	21
4.1.2	int8	21
4.1.3	uint16	22
4.1.4	int16	22
4.1.5	uint32	22
4.1.6	int32	22
4.1.7	char8	22
4.1.8	uchar8	23
4.2	APIs Exported By T.38	24
4.2.1	MT_T38_GetCapabilities	24
4.2.2	MT_T38_SessionStart	25
4.2.3	MT_T38_SessionStop	25
4.2.4	MT_T38_PacketReceived	26
4.2.5	MT_T38_DataRcvdFromDataPump	26
4.2.6	MT_T38_StatusBitsInfo	27
4.2.7	MT_T38_GetT38Statistics	28
4.2.8	MT_T38_TimerCallBack	28
4.3	APIs provided by External Software	29
4.3.1	IFIN_FA_SendPacket	29
4.3.2	IFIN_FA_SendDataToDataPump	29
4.3.3	IFIN_FA_EndOfFAXData	30
4.3.4	IFIN_FA_StartModulator	31
4.3.5	IFIN_FA_StartDemodulator	32
4.3.6	IFIN_FA_DisableDataPump	32
4.3.7	IFIN_IAD_OutTraceString	33
4.4	APIs exported by the Memory Library	33
4.4.1	IFIN_MLIB_Init	33
4.4.2	IFIN_MLIB_AllocMem	34
4.4.3	IFIN_MLIB_FreeMem	34
4.4.4	IFIN_MLIB_Free	35
4.5	APIs exported by the Timer Library	35

CONFIDENTIAL

4.5.1	IFIN_TLIB_StartTimer	35
4.5.2	IFIN_TLIB_StopTimer	36
4.6	Structures	36
4.6.1	x_MT_T38_Caps	38
4.6.2	x_MT_T38_statistics_t	38
4.6.3	x_MT_T38_InitParam	40
4.7	Enumerations	41
4.7.1	e_MT_T38_CapType	41
5	T.38 FAX-Relay Features	42
5.1	Supported FAX Modem Standards	42
5.2	FAX Tone Support	42
5.3	Supported FAX Standards	43
5.4	T.38 Protocol Stack Overview	43
5.5	Additional T.38 Features	44
5.6	QoS Support	44
5.7	Supported T.38 Statistics	44
5.8	Unsupported T.38 Features	45
6	Source Code Organization	46
	References	51

CONFIDENTIAL

List of Figures

Figure 1	Facsimile Transmission over IP Network	10
Figure 2	Software Architecture of T.38 Protocol Stack	12
Figure 3	HDLC Module	13
Figure 4	T.38 Session Flow	20
Figure 5	Source Code Organization of HDLC	46
Figure 6	Source Code Organization of OSIF	46
Figure 7	Source Code Organization of RTL	47
Figure 8	Source Code Organization of T.38	48
Figure 9	Source Code Organization of T38WRP	49
Figure 10	Source Code Organization of COMMON	50

CONFIDENTIAL**List of Tables**

Table 1	Abbreviations	8
Table 2	Structures	36
Table 3	Enumerations	41
Table 4	FAX Modem Standard Table	42
Table 5	FAX Tone Support Table	42
Table 6	Supported FAX Standard Table	43

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.

License Agreement

Infineon T.38 is licensed from the third-party Metasoft. Metasoft has several million copies of FAX software running in the field on a worldwide basis. To use T.38 software a Software License Agreement with Infineon Technologies (covered in Patent Indemnification Agreement) has to be signed.

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 T.38 Protocol Stack interface and the features supported by the T.38 Protocol Stack. The document is written for developers who are integrating and configuring the T.38 Protocol Stack. How to set up the T.38 FAX Relay system integration is described more in detail in the "T.38 Fax Agent Release" document.

Abbreviations

Table 1 Abbreviations

Abbreviation	Full Form
FEC	Forward Error Correction
ASN	Abstract Syntax Notation
TCP	Transmission Control Protocol
UDP	User Data gram Protocol
UDPTL	Facsimile UDP Transport Layer
IFP	Internet Facsimile Protocol
G3FE	Group 3 Facsimile Equipment
CED	Called Terminal Identification
CNG	Calling Tone
DIS	Digital Identification Signal

1 Introduction

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

The T.38 Protocol defines the direct communication between an emitting T.38 and a receiving T.38 gateway. The T.38 data is packetized and wrapped according to the T.38 standard in IFP data packets. The encoding mechanism of the packets and the switch from Voice to FAX is in detail defined within the T.38 standard. Details for working under SIP, H323 and Media Gateways (MG) are defined in the Annexes of the T.38 specification. The mechanism of how a voice gateway informs the receiving voice gateway and how to switch from Voice data to FAX data is defined in the T.38 Annex B for H323, Annex D is used for SIP and Annex E describes the Media Gateways.

The T.38 Protocol Stack implementation is split into two software parts: the FAX Data Pump and the upper layer software - i.e. the T.38 Protocol Stack. The T.38 Protocol Stack is running on the Network Controller and the FAX Data Pump is running on the VINETIC chip set.

For integrating the T.38 Protocol Stack additional software is required which interfaces the T.38 Protocol Stack to the Network Sockets, the VINETIC Telephony Application (TAPI) and Signaling Protocol Stack. The software is called "T.38 FAX Agent" and is part of the T.38 FAX Relay Package. The T.38 FAX Agent is described more in detail in the "T.38 Fax Agent" document.

2 T.38 Protocol Stack

The T.38 Protocol Stack implements algorithms for real-time G3FE facsimile communication over IP networks and is compliant to T.38 ITU-T Recommendation. This chapter describes the T.38 Protocol Stack which is running on the application layer.

The T.30 protocol is running on the G3FE terminal equipment. A subset of the T.30 functionality is implemented in the T.38 Protocol Stack to communicate with the G3FE terminals. The T.38 protocol is used for handling the communication and the FAX data transfer between the T.38 gateways.

The emitting gateway is transmitting the received T.30 data from "Calling" G3FE terminal to the receiving gateway. The receiving gateway is making a PSTN call to the "Called" G3FE terminal. The T.30 session establishment is done between the G3FE terminals and the gateways.

The emitting gateway demodulates the T.30 data from the connected G3FE terminal. Then the emitting gateway sends the T.30 data by using the Internet Facsimile Protocol (IFP) packets over the transport protocol (TCP or UDP) to the receiving gateway.

The G3FE terminal is identified by the FAX tones (CED/CNG/DIS). The FAX tones used for the FAX terminal detection are generated and handled locally between the gateway and the G3FE terminal.

The receiving gateway decodes the received information and establishes the communication with the called G3FE terminal using the T.30 procedures. The receiving gateway forwards the responses from the called G3FE equipment to the emitting gateway.

Figure 1 shows a model for facsimile transmission over IP Networks.

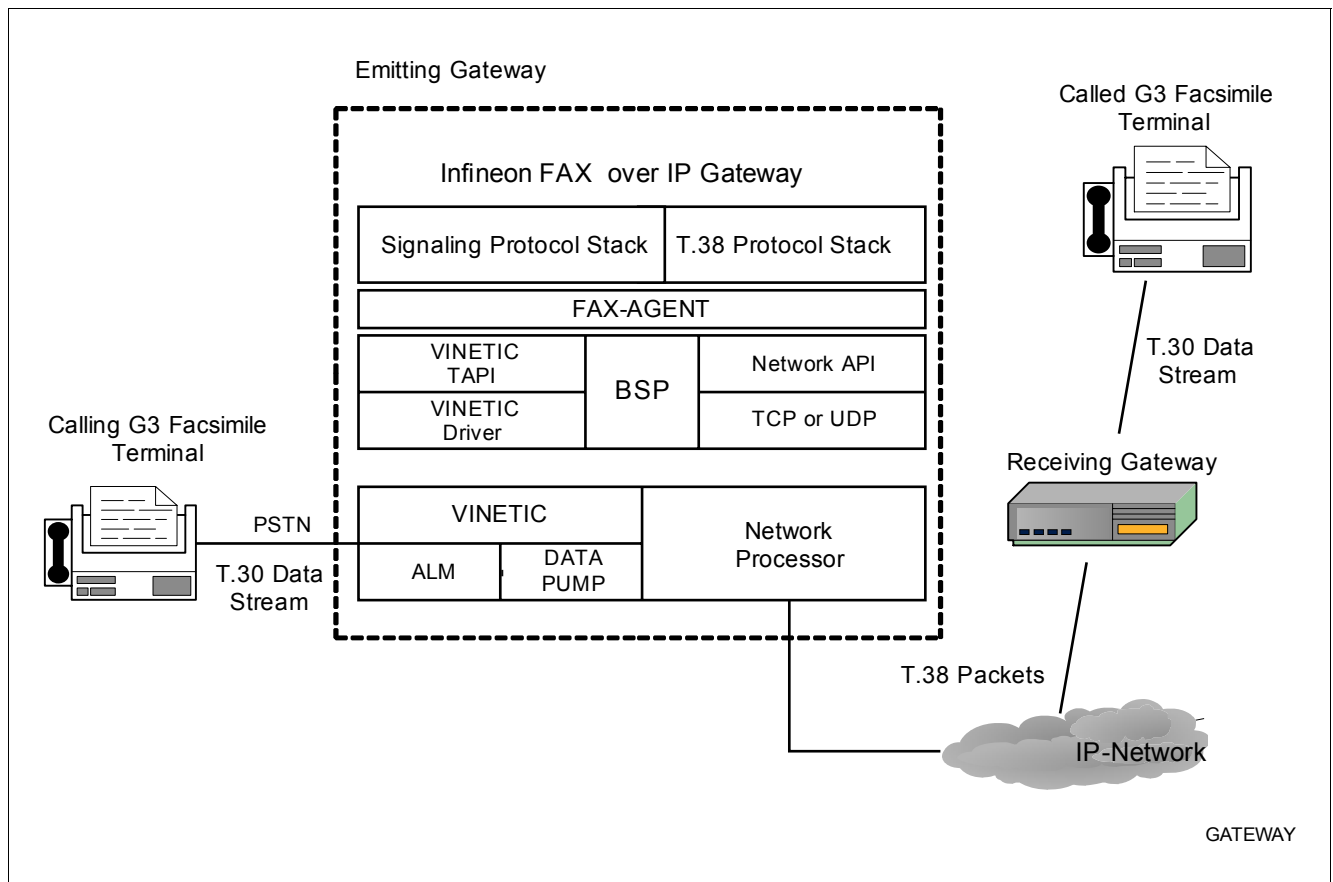


Figure 1 Facsimile Transmission over IP Network

2.1 Modules of T.38 Protocol Stack

The following section gives an overview of the different modules of the T.38 Protocol Stack. The Protocol Stack provides a function based interfaces to the external modules (T.38 FAX Agent, Timer and Memory Library).

Except from the OSIF module which is used for debugging, the T.38 Protocol Stack is a Operating System and platform independent implementation.

The T.38 Protocol Stack is multi channel capable and the number of usable FAX channels depends on the number of supported FAX DATA Pump channels by the VINETIC chip. The multi channel capability depends also on the availability of timers and memory for the FAX channels.

External Module - T.38 FAX Agent

- The T.38 FAX Agent implements the interface which is used by the T.38 Protocol Stack for the FAX Data Packet handling.

External Module - Memory Library

- The T.38 Protocol Stack requires external functionality to allocate memory for the FAX channels, the FAX Data Pump messages and the network data packets. This functionality is implemented by a Memory Library. The Memory Library is not part of the T.38 FAX Relay Package because the Memory Library functionality is system dependent and is different on each system.

External Module - Timer Library

- The T.38 Protocol stack requires external timer functionality to set up a timer for each FAX channel. This functionality is implemented by a Timer Library. The Timer Library is not part of the T.38 FAX Relay Package because the Timer Library is system dependent and is different on each system.

Figure 2 gives an overview of the software architecture for the T.38 Protocol Stack software running on the application level.

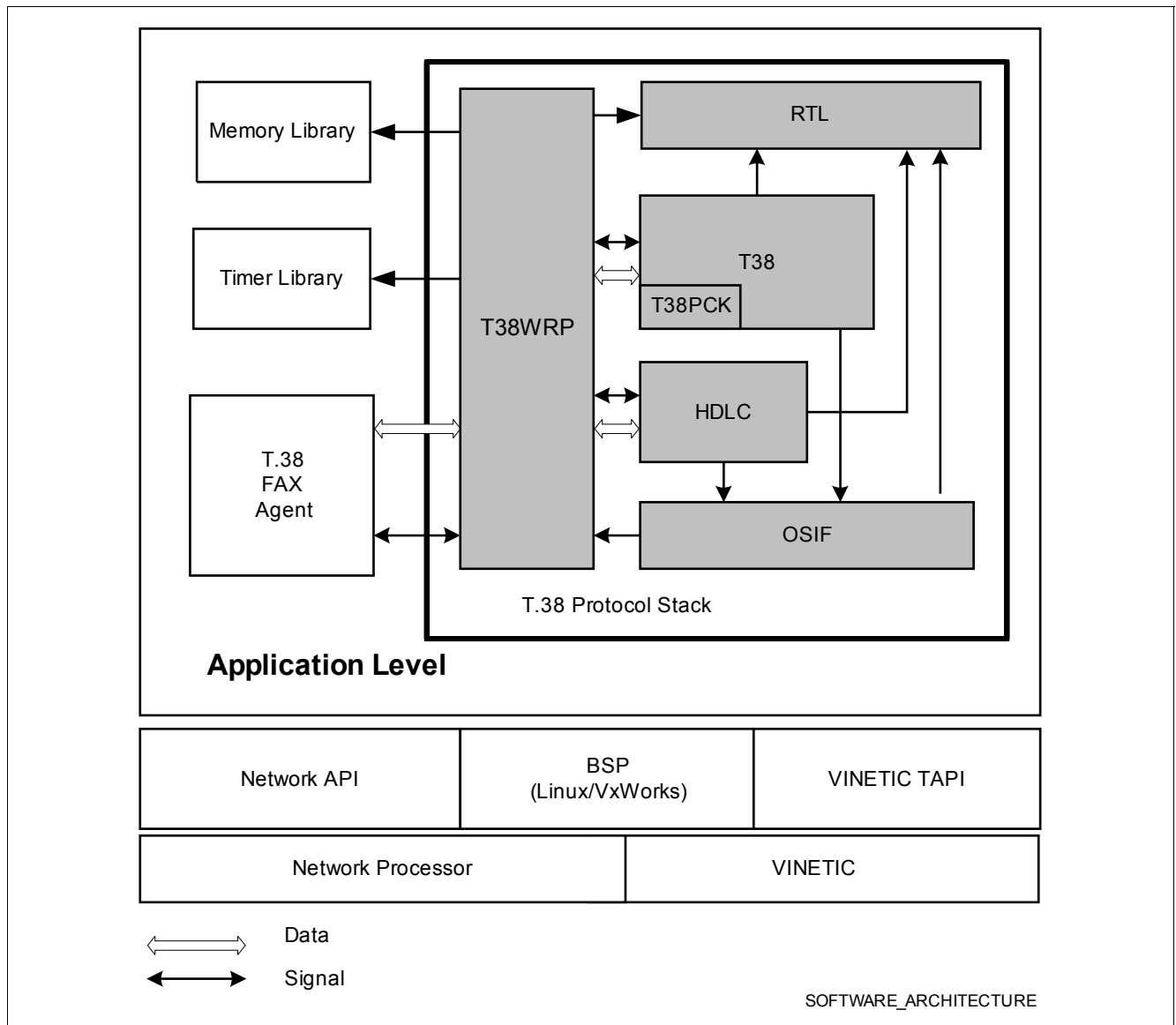


Figure 2 Software Architecture of T.38 Protocol Stack

2.1.1 HDLC Module

The High-level Data Link Control (HDLC) module converts data from and to the serial HDLC bit-stream. All binary coded T.30 facsimile control procedures from and to the G3FE terminal are based on a the HDLC format.

The HDLC frames consist of a number of frames of indeterminable length. The HDLC Module supports error checking and confirmation of correct received T.30 information. The following figure gives an overview of the HDLC data stream.

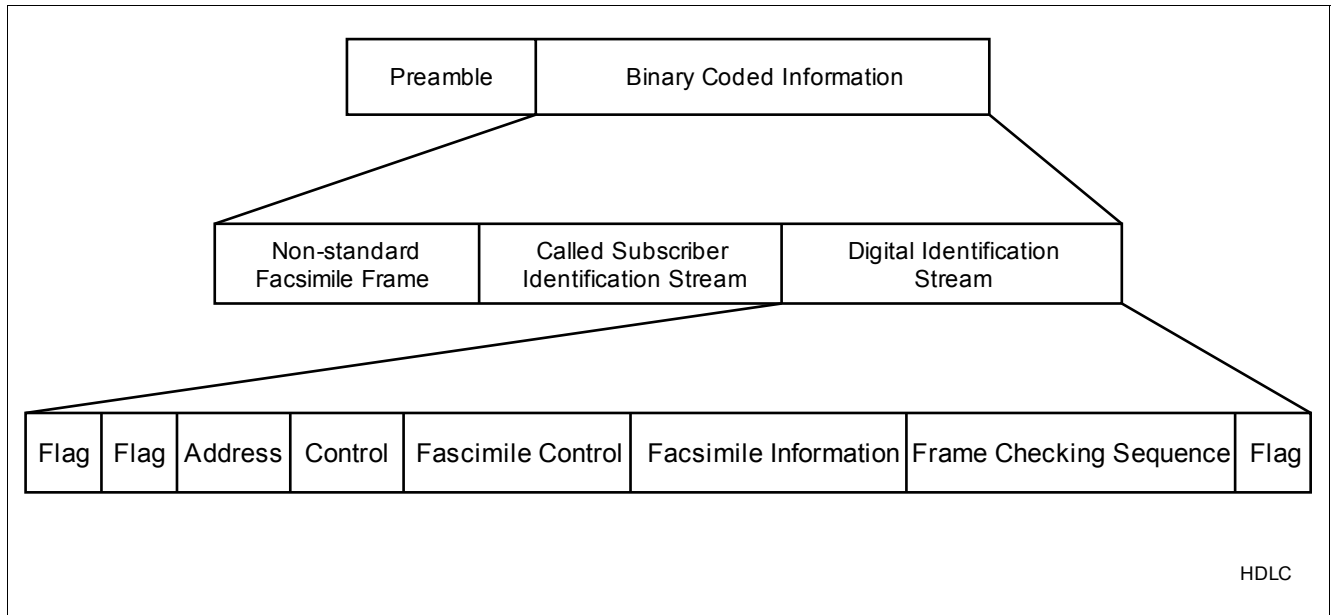


Figure 3 HDLC Module

The HDLC Preamble precedes all binary coded information and is used for the synchronization and sent with each HDLC frame. The Non-standard Facsimile Frame contains all non-standard coded information and handles special features of the G3FE terminal. The Called Subscriber Identification Stream contains the phone number of the G3FE terminal. The Digital Identification Stream contains the HDLC information described in the ITU-T T.30 specification.

The following fields are part of the HDLC frame:

- **FLAG:** This 8-Bit flag sequence is to identify the beginning and end of a frame. For the facsimile transmission the flag is used for bit and frame synchronization. The format of the Flag is 0111 1110.
- **Address Field:** The 8-Bit HDLC address field is used to provide special identification of the specific terminal. In case of the transmission over the PSTN the field format is 1111 1111.
- **Control Field:** The 8-Bit HDLC control field provides the capability of encoding the commands and responses unique to the facsimile procedure. The frame format is 1100 X000, where X=0 is used for non final frames within the procedure. X=1 is used for the final frame within the procedure. A final frame is defined as the last frame transmitted prior to an expected response from the receiving terminal. Therefore intermediate frames have the 1100 0000 bit sequence and final frames have the 1100 1000 bit sequence.
- **Facsimile Information & Control Field:** The HDLC information field is split for T.30 in the Facsimile Control Filed (FCF) and the Facsimile Information Filed (FIF) and is of variable length. The Facsimile Control Field contains the type of information regarding data being exchanged between the two G3FE terminals (DIS, DTC, DCS). The Facsimile Information field contains additional information for the type information defined in the Facsimile Control Field.
- **Frame Checking Sequence:** The FCS frame contains the CRC-16 standard checker number.

In the transmit direction the HDLC Module receives the FAX Data Pump package, converts the HDLC data and removes the bit stuffing to get the HDLC control data and handles the Error Correction Mode (ECM).

In the receive direction the HDLC Module is doing the bit stuffing and converts the FAX data to serial HDLC data, inserts the check sum and handles the Error Correction Mode.

2.1.2 OSIF Module

The Operation System Interface (OSIF) module is used for generating trace information for debug purpose but the OSIF functionality is not required by T.38. The T.38 trace facility can be enabled by a compile switch. If the compile switch is enabled the module initializes a circular buffer of 2 Mbyte and writes the logs from the T.38 Protocol Stack

modules to the circular buffer. During the FAX call the debug information is written to the circular buffer and is flashed to a debug file after the call is finished. The circular debug buffer is used to prevent real time violations during the T.38 call. The OISF Module uses system OS dependent calls and is the only component of the T.38 Protocol Stack which is OS dependent.

2.1.3 RTL Module

The Run Time Library (RTL) for the T.38 Protocol stack implements functionality for buffer handling and CRC calculation. It includes also the spoofing algorithm's functionality of searching the appropriate point for spoofing in one-dimensional and two-dimensional T.30 data. The Run Time Library is not to be understood as a library used by the Operating System.

2.1.4 T38 Module

The T38 Module implements the T.38 Protocol Stack functionality. The capabilities used for the T.38 session are negotiated during the signalling phase by the signaling protocol. The Module controls the demodulation of T.30 signals from the connected G3FE terminal equipment and translates the signals to T.38 Internet Facsimile Protocol (IFP) packets. In the receiving direction the Module translates received T.38 IFP packets to T.30 signals and controls the modulation.

2.1.5 T38WRP Module

The T.38 Wrapper (T38WRP) Module is the "Main Shell" within the T.38 Protocol Stack and provides the T.38 Protocol Stack interface.

The T38WRP Module requests memory for the FAX channels by calling the Memory Library and set up the required timers for FAX channels by calling the Timer Library.

The module controls the data flow between the HDLC Module and the T38 Module and handles the received FAX Data Pump status messages from the T.38 FAX Agent.

2.2 Data Pump

The FAX Data Pump is part of the T.38 FAX Relay package and is running on the VINETIC chip. The FAX Data Pump is described together with the T.38 Protocol Stack because the FAX Data Pump is mandatory to provide the full T.38 FAX Relay functionality.

The FAX traffic on the PSTN consists of modulated digital data. There are several FAX standards using different modulation techniques like: FSK - frequency shift keying (V.21), PSK - phase shifting keying (V.27), QAM - quadrature modulation (V.29, V.17).

The FAX Data Pump implements a FAX Data Pump Modulator for sending the FAX data to the G3FE terminal and a FAX Data Pump Demodulator for the receiving signals from the G3FE terminal.

The T.38 FAX Agent handles the data and signal flow of the FAX Data Pump on the application level. Output data from the demodulator, input data for the modulator and status information is exchanged via data packets.

A T.30 connection is initialized by sending the Calling Tone (CNG). The answering side sends then the Called Terminal Identification Answerer Tone (CED). The CED Tone is followed by the Digital Identification Signal (DIS) to confirm the FAX connection. For detecting the CNG tone the Universal Tone Detector (UTD) is used. For the ANSam and DIS signal the Answering Tone Detector (ATD) is typically used. The CNG and CED Tones are generated by the FAX Data Pump Modulator.

In case of modulation (V.17, V.21, V.27ter, V.29) the controller has to send the FAX data packets to the FAX Data Pump (Downstream direction) over the VINETIC Data Mailbox. The data is stored in the FAX Data Pump input buffer. The buffer size of the input buffer is fixed and is sufficient to hold up to 200msec data for the worst bit rate of 14.400 bit/sec.

In case of demodulation the network controller reads the FAX data packets from the VINETIC Data Mailbox interface. The FAX Data Pump output buffer is fixed and is sufficient to hold up to 95msec data for the worst case bit rate of 14.400 bit/sec.

The FAX Data Pump provides status information over two status bits. One status bit which indicates the controller an error (FDP-ERR bit) and a second status bit (FDP-REQ bit) which requests more data for the modulator.

More detailed information about the FAX Data Pump can be found in the "EDSP Firmware Description" document.

3 T.38 Configuration Parameters

This chapter describes the configuration parameters for the T.38 Protocol Stack and the FAX Data Pump. The T.38 session parameters are defined in Annex B of the T.38 ITU-T recommendation.

3.1 T.38 Session Parameters for Capability Negotiation

There are several options that need to be negotiated to determine which options the gateway supports. The following parameters have to be configured for each T.38 session and have to be supplied to the T.38 Protocol Stack during the start of the T.38 session.

The data `x_MT_T38_Caps` structure is used for the T.38 session parameters and is defined in the file `MT_T38_caps.h`:

- **Data Transport Protocol** - `uiTransportProtocol` - The IFP data packet transport can be done over UDP or TCP. The emitting gateway indicates a preference for UDP over IP or TCP over IP for the transport of T.38 IFP data packets. The receiving gateway selects the transport protocol.
- **Data Rate Management Method** - `ucRateManagement` -
 - Method 1: Method 1 requires that the TCF training signal is generated locally by the receiving gateway. The data rate management is done by the emitting gateway based of the training results from both PSTN connections. Method 1 is used for TCP connections and is optional for UDP.
 - Method 2: Method 2 requires that the TCF signal is transferred from the emitting gateway to the receiving gateway. In this case the speed selection is done by the G3FEs in the same way as they would be on a PSTN connection. Method 2 is mandatory for UDP and not recommended for TCP connections.
- **Fill Bit Removal** - `uiBitOptions` - Indicates the capability to remove and insert fill bits in the Message Transmission Phase C, non-ECM data to reduce bandwidth in the packet network - Optional.
- **MMR Transcoding** - `uiBitOptions` - MMR trans-coding indicates the ability to convert to/from MMR from/to the line format for increasing the compression of the data and reducing the bandwidth in the packet network - Optional.
- **JBIG Transcoding** - `uiBitOptions` - JBIG transcoding indicates the ability to convert to/from JBIG to reduce bandwidth - Optional.
- **Maximum Buffer Size** - `unUDPMaxBufferSize` - For the DP (UDPTL or RTP) mode, this option indicates the maximum number of octets that can be stored on the remote device before an overflow condition occurs. The transmitting application has the responsibility to limit the transfer rate to prevent an overflow.
- **Maximum Datagram Size** - `unUDPMaxDatagramSize` - This option indicates the maximum size of a UDPTL packet or the maximum size of the payload within an RTP packet that can be accepted by the remote side.
- **Version** - The T.38 version number is a mandatory attribute which is to be exchanged between the emitting and the receiving gateway.
- **Error Correction** - `ucUDPErrCorrection` -
 - Supported Error Correction Mode: Redundancy
 - Supported Error Correction Mode: Forward Error Correction (FEC)
- **Maximum Bit Rate** - `unMaxBitRate` - This parameter indicates the maximum permissible bit rate for T.38 connections. The user application can set Maximum Bit Rate bit rate with this parameter for the T.38 session.

3.2 T.38 Protocol Stack Parameters

This section describes the configuration parameters for the T.38 protocol stack which have to be supplied to the T.38 Protocol stack during the start of the T.38 session.

The data structure used for the T.38 Stack parameters is `x_MT_T38_InitParam` and is defined in the file `MT_T38_Interface_Import.h`.

- **Options** - `iOptions` - This parameter indicates the configurable options for the T.38 protocol stack and a combination of the options is possible:

- **NSX Patch** - MT_T38_INIT_OPT_NSXPATCH - There are two methods for handling non standard facsimiles. One is to discard the non standard frames and the second one is to patch the non standard information. This option enables the patching of the non standard facsimile information.
- **ASN.1 Version** - MT_T38_INIT_OPT_USE_OLD_ASN98 - Setting this option switches from the ASN.1 2002 notification to the ASN.1 1998 notification. If gateways using the ASN.1 notification 1998 it is recommended to set this option otherwise it is not possible to establish the T.38 session.
- **Server Impairments** - MT_T38_INIT_OPT_SEVERE_IP_IMPAIRMENTS - Enables the support of dealing with extended round trip delays.
- **Calling Gateway** - MT_T38_INIT_OPT_CALLING - Configures the gateway as the calling gateway.
- **Data Wait Time** - uiDataWaitTime - This parameter indicates the time for buffering appropriate V.21, ECM and non-ECM page data in the case if the end of line or the end of the HDLC frame is not detected. The value range is from 0 up to 100. The default value is to set to 50 which is of 500 ms buffering. One unit is of 10 ms. After the Data Wait Time T.38 starts with the modulation also if not all data is received.
- **High Rate Packet Recovery** - uiUdpHighRateErrRecoveryPackets - This parameter is used for the Redundancy and the FEC error Correction Mode on a UDP transmission for V.17, V.29, and V.27. The parameter indicates the number of additional recovery data packets send during a high rate FAX transmission (image data).
- **Low Rate Packet Recovery** - uiUdpLowRateErrRecoveryPackets - This parameter is used for the Redundancy and the FEC error Correction Mode on a UDP transmission for V.21. The parameter indicates the number of additional recovery data packets send during a low rate FAX transmission (image data).
- **UDP Packets for FEC** - uiUdpPriorPacketsForFEC - This parameter is only applicable for the Forward Error Correction Mode (FEC) on a UDP transmission and defines the number of packets for calculating the FEC mode.
- **NSX Information Field Size** - uiNsxInfoFieldSz - Configures the size of the NSX Information Field. Some FAX machine manufactures developed proprietary FAX protocols different from the T.30 standard and also different to the modulation methods (V.21, V.17, V.27, V.29). These protocols are activated when both FAX devices recognize that they belong to the same manufacturer. T.38 can not support proprietary protocols and has to remove these fields to prevent the non standard recognition or to patch the NSF, NSS and NSC by defined. This mechanism is called NSX patching. The NSX patch can be activated with the Options settings.
- **NSX Information Field** - *pucNsxInfoField - This parameter is used to configure the NSX patch by predefined values. The first bit transmitted should be stored in the Most Significant Bit (MSB) of the first octet. Usage of long frames increases the delay of the FAX handshaking mechanism. This parameter is applicable if the NSX patch is enabled in the Options settings.
- **Waiting Time** - uiAutoStartWaitTime - Defines the wait time for the response time of commands. If the command response is not received within the Waiting Time the T.38 stack starts with modulation to prevent real time violations.

3.3 Data Pump Parameters

The following parameters are required by the T.38 Protocol Stack to configure the T.38 FAX Data Pump and have to be supplied to the T.38 Protocol Stack during the start of the T.38 session.

The data structure used for the Data Pump is **x_MT_T38_InitParam** and is defined in the file **MT_T38_InterfaceImport.h**.

- **Gain Values** - The parameters configures the transmission and reception gain values for the FAX Data Pump:
 - **unGain1** - The Gain1 value is the gain for upstream (demodulation) direction of the FAX Data Pump.
 - **unGain2** - The Gain2 value is the gain for downstream (modulation) direction of the FAX Data Pump.
- **Signal Length** - The signal duration in milliseconds is used for the tonal signal generation of CNG, CED and also for generating silence.
- **DBM Level** - uiDBmLevel - The DBM value is the desired output signal power in -dBm for the FAX Modulator. The meaning of “-dBm” is that positive values have to be used instead of negative ones (dBm value is negative by definition). For example a DBM value of 10 means desired output signal power equal to -10 dBm.

- **Data Pump Resource Number** - $ucDPNR$ - Each FAX Data Pump channel has to be configured with a Data Pump resource number. The Data Pump resources are overlaid with the coder resources that means the same resource number mustn't be used for a FAX Data Pump channel and a Voice Coder channel.
- **MOBSM** - $unMOBSM$ - $MOBSM = MOBSM [msec] / 0.625 msec$, $0.625 \leq MOBSM \leq 200ms$. This value defines the level for starting the modulation from the modulation buffer.
- **MOBRD** - $unMOBRD$ - $MOBRD = MOBRD [msec] / 0.625 msec$, $0.625 \leq MOBRD \leq 200ms$. This value defines the level for requesting more data for modulation from the controller. The packet size in downstream direction is limited to 200ms minus MOBRD because 200ms is the maximum buffer size and the controller gets a request if the data within the FAX Data Pump input buffer represents a time frame below MOBRD.
- **DMBSD** - $unDMBSD$ - $MOBSD = MOBSD [msec] / 0.625 msec$, $0.625 \leq MOBSD \leq 95ms$. This value defines the send level for the demodulation output buffer. If the number of bytes within the FAX Data Pump output buffer represents at least DMBSD msec the data packet is send to the controller.

Notes

1. *MOBSM, MOBRD and DMBSD are defined in time units to make the buffering independent from the bit rate.*
2. *A detailed description of the Data Pump and its parameters and setting can be found in the EDSP Firmware Description.*

4 Interface Description

The following section describes the T.38 Protocol Stack interface. The T.38 Protocol Stack interface is a function based interface to the T.38 FAX Agent, the Memory Library and to the Timer Library.

Functions provided by the T.38 Protocol are used by the T.38 FAX Agent and the functions provided by the T.38 FAX Agent are used by the T.38 Protocol stack.

The timer and memory functionality required by the T.38 Protocol Stack has to be provided by the Timer Library and the Memory Library.

Figure 4 illustrates the T.38 function calls during a T.38 session and shows the flow of the function calls.

Functions exported by the T.38 Protocol Stack

Functions implemented inside the T.38 Protocol Stack and provided by the T.38 Protocol Stack are preceded by the prefix MT_T38_. These functions are called by the external modules.

Functions provided by the external code

The functions provided by the external code are preceded by the prefix IFIN_. These functions are called by the T.38 Protocol Stack.

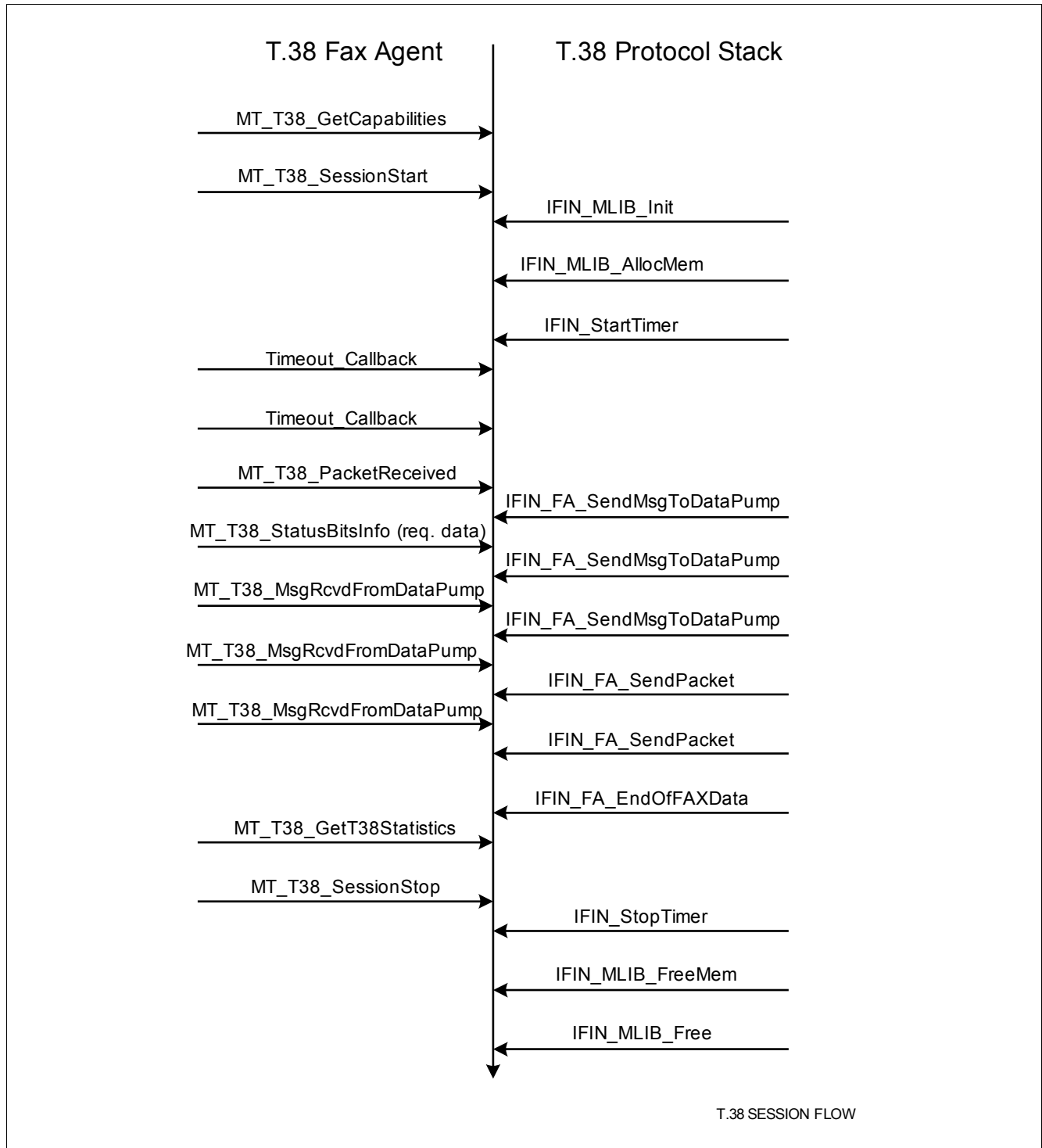


Figure 4 T.38 Session Flow

4.1 Basic Type Definitions

This section describes the basic type definitions used by the T.38 Protocol Stack.

- [uint8](#)
- [int8](#)
- [uint16](#)
- [int16](#)
- [uint32](#)
- [int32](#)
- [char8](#)
- [uchar8](#)

4.1.1 uint8

Prototype

```
typedef unsigned char uint8;
```

Parameters

Data Type	Name	Description
unsigned char	uint8	Unsigned char 8-bit datatype.

4.1.2 int8

Prototype

```
typedef char int8;
```

Parameters

Data Type	Name	Description
char	int8	Char 8-bit datatype.

4.1.3 uint16

Prototype

```
typedef unsigned short int uint16;
```

Parameters

Data Type	Name	Description
unsigned short int	uint16	Unsigned short integer 16-bit datatype.

4.1.4 int16

Prototype

```
typedef short int int16;
```

Parameters

Data Type	Name	Description
short int	int16	Signed short integer 16-bit datatype.

4.1.5 uint32

Prototype

```
typedef unsigned int uint32;
```

Parameters

Data Type	Name	Description
unsigned int	uint32	Unsigned int 32-bit datatype.

4.1.6 int32

Prototype

```
typedef int int32;
```

Parameters

Data Type	Name	Description
int	int32	Int 32-bit datatype.

4.1.7 char8

Prototype

```
typedef char char8;
```

CONFIDENTIAL

Interface Description

Parameters

Data Type	Name	Description
char	char8	Char 8-bit datatype.

4.1.8 uchar8**Prototype**

```
typedef unsigned char uchar8;
```

Parameters

Data Type	Name	Description
unsigned char	uchar8	Unsigned char 8-bit datatype.

4.2 APIs Exported By T.38

The following sub chapter describes the APIs provided by the T.38 Protocol Stack. The APIs are called by the T.38 FAX Agent.

4.2.1 MT_T38_GetCapabilities

Description

The MT_T38_GetCapabilities() API is called to obtain the T.38 Protocol Stack capabilities and the function returns the capabilities. The capabilities are used by the signaling entity (SIP, H323) to announce the T.38 capabilities to the peer gateway. By using this information the signaling entity is able to create one or several T.38 sessions. For example the first T.38 session is using TCP and Rate Management Method 1 and the second T.38 session is using UDP and Rate Management Method 2.

Prototype

```
int32 MT_T38_GetCapabilities(
    e_MT_T38_CapType eCapabilitiesType,
    x_MT_T38_Caps *pxCapabilities);
```

Parameters

Data Type	Name	Description
e_MT_T38_CapType	eCapabilitiesType	<ul style="list-style-type: none"> MT_T38_CapType_TCP - TCP capabilities MT_T38_CapType_UDP - UDP capabilities
x_MT_T38_Caps	*pxCapabilities	Pointer to structure containing T.38 Protocol capabilities.

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> MT_T38_SUCCESS, on success MT_T38_FAILURE, on failure

4.2.2 MT_T38_SessionStart

Description

The MT_T38_SessionStart() API is used by the T.38 FAX Agent and starts the T.38 Session by opening the FAX channel, allocating the memory for the channel and configuring the parameters for the T.38 session. The API is to call before calling any other T.38 API service (except MT_T38_GetCapabilities). The T.38 FAX Agent is calling the MT_T38_SessionStart API after the setup of the T.38 connection to the remote side is done.

Prototype

```
int32 MT_T38_SessionStart(
    uint16          unChannelID,
    x_MT_T38_InitParam *pxInitParams,
    x_MT_T38_Caps    *pxSessionCapabilities,
    IFIN_MLIB_Id     *pMlibId);
```

Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.
x_MT_T38_InitParam	*pxInitParams	Pointer to the T.38 Protocol Stack Parameters.
x_MT_T38_Caps	*pxSessionCapabilities	Pointer to the T.38 Capability Session Parameters.
IFIN_MLIB_Id	*pMlibId	Pointer to the allocated memory for the FAX channels.

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> MT_T38_SUCCESS, on success MT_T38_FAILURE, on failure

4.2.3 MT_T38_SessionStop

Description

The MT_T38_sessionStop() API is used by the T.38 FAX Agent and stops the T.38 session, clears the parameters and deallocates the FAX channel memory.

Prototype

```
int32 MT_T38_SessionStop(uint16 unChannelID);
```

Parameters

Data Type	Name	Description
uint16	unChannelID	Channel hosting fax.

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> MT_T38_SUCCESS, on success MT_T38_FAILURE, on failure

4.2.4 MT_T38_PacketReceived

Description

The MT_T38_PacketReceived() API is used by the T.38 FAX Agent and sends the TCP/UDP packet payload received from the network to the T.38 Protocol Stack.

Prototype

```
int32 MT_T38_PacketReceived(
    uint16 unChannelID,
    uchar8 *pucInBuffer,
    uint32 uiInBufferSize);
```

Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.
uchar8	*pucInBuffer	Pointer to the TCP or UDP payload data.
uint32	uiInBufferSize	Packet data size in bytes. Max Packet Size is of 2412 bytes.

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> MT_T38_SUCCESS, on success MT_T38_FAILURE, on failure

4.2.5 MT_T38_DataRcvdFromDataPump

Description

The MT_T38_DataRcvdFromDataPump() API is used by the T.38 FAX Agent and sends the FAX Data Pump packet to the T.38 Protocol stack.

Prototype

```
int32 MT_T38_DataRcvdFromDataPump(
    uint16 unChannelID,
    uchar8 *pucDataPumpMsg,
    uint32 uiMsgSize);
```

CONFIDENTIAL
Interface Description
Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.
uchar8	*pucDataPumpMsg	Pointer to the FAX Data Pump packet.data received from Datapump
uint32	uiMsgSize	Packet data size in bytes. Max size of data is of 514 bytes.

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> MT_T38_SUCCESS, on Success MT_T38_FAILURE, on Failure

4.2.6 MT_T38_StatusBitsInfo

Description

The MT_T38_StatusBitsInfo() API is used by the T.38 FAX Agent to forward the status bit information from the FAX Data Pump to the T.38 Protocol Stack. If the FAX Data Pump requires more data for modulation, the FAX Data Pump Request Bit (FDP_REQ) is send to the controller.

The T.38 FAX Agent forwards the request to the T.38 Protocol Stack. Then the T.38 Protocol Stack sends the requested data to the FAX Data Pump modulator. In case of an FAX Data Pump error the FAX Data Pump send the FAX Data Pump Error bit (FDP_ERR) to the controller. The T.38 FAX Agent forwards the error bit to the T.38 Protocol Stack and the stack closes the T.38 session in case of an FDP_ERR.

Prototype

```
int32 MT_T38_StatusBitsInfo(
    uint16 unChannelID,
    int8 iStatusBit);
```

Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.
int8	iStatusBit	<ul style="list-style-type: none"> MT_DSP_FDP_REQ_BIT - Request more data for modulation MT_DSP_FDP_ERR_BIT - Report DSP error

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> MT_T38_SUCCESS, on success MT_T38_FAILURE, on failure

4.2.7 MT_T38_GetT38Statistics

Description

The MT_T38_GetT38Statistics() API is used by the T.38 FAX Agent to request the statistics of an on going T.38 session from the T.38 Protocol Stack.

Prototype

```
int32 MT_T38_GetT38Statistics(
    uint16 unChannelID,
    x_MT_T38_statistics_t *px_T38Statistics);
```

Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.
x_MT_T38_statistics_t	*px_T38Statistics	Pointer to the structure containing the statistic information.

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> MT_T38_SUCCESS, on success MT_T38_FAILURE, on failure

4.2.8 MT_T38_TimerCallback

Description

The MT_T38_TimerCallback() API is used by the T.38 FAX Agent to indicate the time out.

Prototype

```
extern int32 MT_T38_TimerCallback(uint16 unChannelID);
```

Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> MT_T38_SUCCESS, on success MT_T38_FAILURE, on failure

4.3 APIs provided by External Software

The following sub chapter describes the APIs provided by the T.38 FAX Agent. The APIs are called by the T.38 Protocol Stack.

4.3.1 IFIN_FA_SendPacket

Description

The IFIN_FA_SendPacket API is called by the T.38 Protocol Stack for sending a TCP/UDP T.38 data packet to the remote gateway. The T.38 Protocol Stack wraps the received FAX Data Pump data in IFP packets and calls the IFIN_SendPacket API to send the packet to the network.

Prototype

```
int32 IFIN_FA_SendPacket(
    uint16 unChannelID,
    uchar8 *pucOutBuffer,
    uint32 uiOutBufferSize);
```

Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number
uchar8	*pucOutBuffer	Pointer to the IFP data packet.
uint32	uiOutBufferSize	Packet size in bytes

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> IFIN_FA_SUCCESS, on success IFIN_FA_FAILURE, on failure

4.3.2 IFIN_FA_SendDataToDataPump

Description

The IFIN_FA_SendDataToDataPump() API is called by the T.38 Protocol Stack to send data for modulation to the FAX Data Pump.

The IFP data packet received by the remote side is processed by the T.38 Protocol Stack and sent to the FAX Data Pump by calling the IFIN_FA_SendDataToDataPump() API.

Prototype

```
int32 IFIN_FA_SendDataToDataPump(
    uint16 unChannelID,
    uchar8 *pucMsgToDataPump,
    uint32 uiMsgSize);
```

CONFIDENTIAL
Interface Description
Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.
uchar8	*pucMsgToDataPump	Pointer to FAX Data Pump data.
uint32	uiMsgSize	Packet size in bytes. Max size of data is of 360 bytes.

Return Values

Data Type	Description
int32	The return value can be either of the following: <ul style="list-style-type: none"> • IFIN_FA_SUCCESS, on success • IFIN_FA_FAILURE, on failure

4.3.3 IFIN_FA_EndOfFAXData

Description

The IFIN_FA_EndOfFAXData() API is called by the T.38 Protocol Stack to indicate the end of the FAX transmission to the T.38 FAX Agent.

Prototype

```
int32 IFIN_FA_EndOfFAXData(uint16 unChannelID);
```

Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> • IFIN_FA_SUCCESS, on success • IFIN_FA_FAILURE, on failure

4.3.4 IFIN_FA_StartModulator

Description

The IFIN_FA_StartModulator API called by the T.38 Protocol stack starts the FAX Data Pump modulator.

Prototype

```
int32 IFIN_FA_StartModulator(
    uint16 unChannelID,
    uchar8 ucDBM,
    uchar8 ucTEP,
    uchar8 ucTRN,
    uchar8 ucSTD,
    uint16 unSGLEN);
```

Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.
char8	ucDBM	Desired output signal power in -dBm.
uchar8	ucTEP	<ul style="list-style-type: none"> IFIN_FA_NO_TEP IFIN_FA_GENERATE_TEP
uchar8	ucTRN	<ul style="list-style-type: none"> IFIN_FA_SHORT_TRAINING_SEQ IFIN_FA_LONG_TRAINING_SEQ
uchar8	ucSTD	<ul style="list-style-type: none"> Silence V.21 V.27ter / 2400 bps V.27ter / 4800 bps V.29 / 9600 bps V.29 / 7200 bps V.17 / 14400 bps V.17 / 12000 bps V.17 / 9600 bps V.17 / 7200 bps CNG CED
uint16	unSGLEN	Duration of the CED, CNG and silence.

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> IFIN_FA_SUCCESS, on success IFIN_FA_FAILURE, on failure

4.3.5 IFIN_FA_StartDemodulator

Description

The IFIN_FA_StartDemodulator() API is called by the T.38 Protocol Stack to start the FAX Data Pump demodulator.

Prototype

```
int32 IFIN_FA_StartDemodulator(
    uint16 unChannelID,
    uchar8 ucTRN,
    uchar8 ucEQ,
    uchar8 ucSTD2,
    uchar8 ucSTD1);
```

Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.
uchar8	ucTRN	<ul style="list-style-type: none"> IFIN_FA_SHORT_TRAINING_SEQ IFIN_FA_LONG_TRAINING_SEQ
uchar8	ucEQ	<ul style="list-style-type: none"> IFIN_FA_RESET_EQUALIZER_B4_DEMOD IFIN_FA_REUSE_PREV_EQUALIZER_COEF FS
uchar8	ucSTD2	Alternative standard to STD1, if the receiver does not know the standard for demodulation the alternative signal STD2 is supplied to the FAX Data Pump.
uchar8	ucSTD1	Desired standard to be used by the FAX Data Pump.

Return Values

Data Type	Description
int32	The return value can be either of the following: <ul style="list-style-type: none"> IFIN_FA_SUCCESS, on success IFIN_FA_FAILURE, on failure

4.3.6 IFIN_FA_DisableDataPump

Description

The IFIN_FA_DisableDataPump() API is called by the T.38 Protocol Stack to deactivate the FAX Data Pump.

Prototype

```
int32 IFIN_FA_DisableDataPump(uint16 unChannelID);
```


CONFIDENTIAL
Interface Description
Parameters

Data Type	Name	Description
uint16	unChannelID	FAX channel number.

Return Values

Data Type	Description
int32	<ul style="list-style-type: none"> IFIN_FA_SUCCESS, on success IFIN_FA_FAILURE, on failure

4.3.7 IFIN_IAD_OutTraceString

Description

The IFIN_IAD_OutTraceString() API is called by the T.38 Protocol Stack to log the T.38 session traces.

Prototype

```
void IFIN_IAD_OutTraceString(char8 *pcString);
```

Parameters

Data Type	Name	Description
char8	*pcString	Null terminated string

4.4 APIs exported by the Memory Library

The following sub chapter describes the APIs provided by the Memory Library. The APIs are called by the T.38 Protocol Stack for the FAX channel memory allocation.

4.4.1 IFIN_MLIB_Init

Description

The IFIN_MLIB_Init API() is called by the T.38 Protocol Stack to initialize the Memory Library. The IFIN_MLIB_Init() function is setting pointers to data buffers and is setting the current status of the data buffers.

Prototype

```
int16 IFIN_MLIB_Init(
    IFIN_MLIB_Id      *pMlibId,
    uint16            unKeyId,
    uchar8            ucMlibType,
    uchar8            *pucUserMemory,
    x_IFIN_MLIB_SegInfo *pxSegInfo,
    uint32            uiSharedDataSize
)
```

Parameters

Data Type	Name	Description
IFIN_MLIB_Id	*pMlibId	Memory ID
uint16	unKeyId	Key ID
uchar8	ucMmuType	Memory Type
uchar8	*pucUserMemory	User Memory
x_IFIN_MLIB_SegInfo	*pxSegInfo	Segment Info
uint32	uiSharedDataSize	Shard Size

Return Values

Data Type	Description
	<ul style="list-style-type: none"> IFIN_MLIB_SUCCESS IFIN_MLIB_FAILURE

4.4.2 IFIN_MLIB_AllocMem

Description

The IFIN_MLIB_AllocMem() API is called by the T.38 Protocol Stack to allocate memory for the FAX channels.

Prototype

```
uchar8* IFIN_MLIB_AllocMem( IFIN_MLIB_Id MlibId, uint32 uiSegSize )
```

Parameters

Data Type	Name	Description
IFIN_MLIB_Id	MlibId	ID
uint32	uiSegSize	Memory Size

Return Values

Data Type	Description
unsigned char	<ul style="list-style-type: none"> Pointer to allocated memory

4.4.3 IFIN_MLIB_FreeMem

Description

The IFIN_MLIB_FreeMem() API is called by the T.38 Protocol Stack to deallocate the memory.

Prototype

```
int16 IFIN_MLIB_FreeMem(IFIN_MLIB_Id MlibId, uchar8 *pucSegAddr)
```

CONFIDENTIAL
Interface Description
Parameters

Data Type	Name	Description
IFIN_MLIB_Id	MlibId	ID
uchar8	*pucSegAddr	Pointer to memory

Return Values

Data Type	Description
	<ul style="list-style-type: none"> IFIN_MLIB_SUCCESS IFIN_MLIB_FAILURE

4.4.4 IFIN_MLIB_Free
Description

The IFIN_MLIB_Free() API is called by the T.38 Protocol Stack.

Prototype

```
int16 IFIN_MLIB_Free(IFIN_MLIB_Id MlibId)
```

Parameters

Data Type	Name	Description
IFIN_MLIB_Id	MlibId	ID

Return Values

Data Type	Description
	<ul style="list-style-type: none"> IFIN_MLIB_SUCCESS

4.5 APIs exported by the Timer Library

The following sub chapter describes the APIs exported by the Timer Library. The APIs are called by the T.38 Protocol Stack.

4.5.1 IFIN_TLIB_StartTimer
Description

The IFIN_StartTimer() API is called by the T.38 Protocol Stack.

Prototype

```
int8 IFIN_TLIB_StartTimer(
    uint16 *punTimerId,
    uint32 uiTimerValue,
    uint8 ucTimerType,
    pfnVoidFuncPtr pfn_IFIN_TLIB_Callbackfn,
    void *pCallbackFnParm)
```

Parameters

Data Type	Name	Description
uint16	*punTimerID	Timer Id
uint32	uiTimerValue	Timer Value in micro seconds
uint8	ucTimerType	<ul style="list-style-type: none"> IFIN_TLIB_ONE_TIME_TIMER IFIN_TLIB_PERIODIC_TIMER
pfnVoidFuncPtr	pfn_IFIN_TLIB_CallBack fn	Function call back
void	*pCallBackFnParm	Pointer to call back function parameter

Return Values

Data Type	Description
	<ul style="list-style-type: none"> IFIN_TLIB_SUCCESS IFIN_TLIB_FAILURE

4.5.2 IFIN_TLIB_StopTimer

Description

The IFIN_StartTimer is called by the T.38 Protocol stack to start the timer for the FAX channel.

Prototype

```
int8 IFIN_TLIB_StopTimer(uint16 unTimerId)
```

Parameters

Data Type	Name	Description
uint16	unTimerId	Timer Id

Return Values

Data Type	Description
	<ul style="list-style-type: none"> IFIN_TLIB_SUCCESS IFIN_TLIB_FAILURE

4.6 Structures

The following sub chapter describes the definitions and structures listed in the following table.

Table 2 Structures

Name	Description
x_MT_T38_Caps	Defines the parameters used for the T.38 Protocol Stack capability negotiation.

CONFIDENTIAL**Interface Description****Table 2 Structures (cont'd)**

Name	Description
x_MT_T38_InitParam	Defines the partners used for the T.38 Protocol Stack initialization.
x_MT_T38_statistics_t	Defines the parameters for the T.38 Protocol Stack statistics.

4.6.1 x_MT_T38_Caps

Description

The x_MT_T38_Caps structure includes the parameters defined in ITU-T T.38 Annex B specification.

Prototype

```
typedef struct{
    uint32 uiTransportProtocol;
    uchar8 ucVersion;
    uchar8 ucRateManagement;
    uint16 unMaxBitRate;
    uint32 uiBitOptions;
    uint16 unUDPMaxBufferSize;
    uint16 unUDPMaxDatagramSize;
    uchar8 ucUDPErrCorrection;
} x_MT_T38_Caps;
```

Parameters

Data Type	Name	Description
uint32	uiTransportProtocol	e_MT_T38_CapType .
uchar8	ucVersion	T.38 Protocol version
uchar8	ucRateManagement	<ul style="list-style-type: none"> MT_T38_CAPS_FL_TCF_LOCAL - Data Rate Management Method 1. MT_T38_CAPS_FL_TCF_TRANSFERRED - Data Rate Management Method 2.
uint16	unMaxBitRate	Maximum allowed bit rate for the T.38 Session. The parameter is defined in 100 bit/sec. Max value is of 144 which corresponds to 14400 bits/sec. Possible values are 144, 120, 96, 72, 48, 24.
uint32	uiBitOptions	<ul style="list-style-type: none"> MT_T38_CAPS_FL_OPT_FILL_BIT_REMOVAL MT_T38_CAPS_FL_OPT_TRANSCODING_MMR MT_T38_CAPS_FL_OPT_TRANSCODING_JBIG
uint16	unUDPMaxBufferSize	Maximum Buffer Size (required only for UDP). Defines the maximum buffer size for UDP in bytes. The value is obtained during the capability negotiation and is passed to the T.38.
uint16	unUDPMaxDatagramSize	Maximum Datagram Size (required only for UDP). Defines the maximum data gram size in bytes. The value is obtained during the capability negotiation and is passed to the T.38.
uchar8	ucUDPErrCorrection	<ul style="list-style-type: none"> MT_T38_CAPS_FL_EC_REDUNDANCY MT_T38_CAPS_FL_EC_FEC

4.6.2 x_MT_T38_statistics_t

Description

The x_MT_T38_statistics_t definition includes the parameters used for the T.38 Protocol Stack statistics.

CONFIDENTIAL
Interface Description
Prototype

```
typedef struct{
    int32          i_Flags;
    int32          i_Standards;
    int32          ui_RecoveredPackets;
    uint32         ui_MaxLostPcksGroup;
    uint32         ui_StatFTTcount;
    uint32         ui_PagesTransferred;
    uint32         ui_LineBreaks;
    uint32         ui_v21FrmBreaks;
    uint32         ui_ECM_FrmBreaks;
    uint32         ui_MajorVersion;
    uint32         ui_MinorVersion;
} x_MT_T38_statistics_t;
```

Parameters

Data Type	Name	Description
int32	i_Flags	<ul style="list-style-type: none"> MT_T38_ST_FEC_USED - Remote used FEC mode MT_T38_ST_REDUND_USED - Remote used Redundancy mode MT_T38_ST_ECM_PAGE - ECM Page Transaction MT_T38_ST_COMPLETE_TRANSACTION - Transaction Completed
int32	i_Standards	<ul style="list-style-type: none"> MT_T38_ST_STD_V27TER_2400 MT_T38_ST_STD_V27TER_4800 MT_T38_ST_STD_V29_7200 MT_T38_ST_STD_V29_9600 MT_T38_ST_STD_V17_7200 MT_T38_ST_STD_V17_9600 MT_T38_ST_STD_V17_12000 MT_T38_ST_STD_V17_14400
int32	ui_RecoveredPackets	Number of recovered data packets.
uint32	ui_MaxLostPcksGroup	Maximum number of consecutive lost packets.
uint32	ui_StatFTTcount	Number of TCF responded with FTT.
uint32	ui_PagesTransferred	Number of transferred pages.
uint32	ui_LineBreaks	Number of broken non-ECM page lines.
uint32	ui_v21FrmBreaks	Number of broken v21 modulation frames.
uint32	ui_ECM_FrmBreaks	Number of broken ECM modulation frames.
uint32	ui_MajorVersion	Major T38 build version
uint32	ui_MinorVersion	Minor T38 build version

4.6.3 x_MT_T38_InitParam

Description

The x_MT_T38_InitParam structure defines the parameters used for the initialization of the T.38 Protocol stack.

Prototype

```
typedef struct {
    int32  iOptions;
    uint32 uiDBmLevel;
    uint32 uiDataWaitTime;
    uint32 uiUdpHighRateErrRecoveryPackets;
    uint32 uiUdpLowRateErrRecoveryPackets;
    uint32 uiUdpPriorPacketsForFEC;
    uchar8 *pucNsxInfoField;
    uint32 uiNsxInfoFieldSz;
    uint16 unGain1;
    uint16 unGain2;
    uchar8 ucDPNR;
    uchar8 ucInputSignal;
    uchar8 ucDataFrameLength;
    uint16 unMOBSM;
    uint16 unMOBRD;
    uint16 unDMBSD;
    uint32 uiAutoStartWaitTime;
    uint32 uiAutoSpoofingTime;
} x_MT_T38_InitParam;
```

Parameters

Data Type	Name	Description
int32	iOptions	<ul style="list-style-type: none"> MT_T38_INIT_OPT_CALLING MT_T38_INIT_OPT_NSXPATCH MT_T38_INIT_OPT_USE_OLD_ASN98 MT_T38_INIT_OPT_SEVERE_IP_IMPAIRMENTS
uint32	uiDBmLevel	Output voice power level in dBm
uint32	uiDataWaitTime	Time for buffering appropriate V.21, ECM and non-ECM page data.
uint32	uiUdpHighRateErrRecoveryPackets	Number of additional data recovery packets.
uint32	uiUdpLowRateErrRecoveryPackets	Number of additional data recovery packets.
uint32	uiUdpIndicatorDuplicationPackets	Duplication of T.30 indicator.
uint32	uiUdpPriorPacketsForFEC	Number of packets to calculate the FEC.
char8	*pucNsxInfoField	Used for NSX patching.
uint32	uiNsxInfoFieldSz	Number of data bytes used for the NSX patch.
uint16	unGain1	Gain in upstream direction.

Data Type	Name	Description
uchar8	ucGain2	Gain in downstream direction.
char8	ucDPNR	Data pump resource number.
char8	ucInputSignal	Input signal to the FAX Data Pump.
uint16	unMOBSM	Modulation buffer, level for start modulation.
uint16	unMOBRD	Modulation buffer, data request level.
uint16	unDMBSD	Demodulation buffer, data send level.
int32	uiAutoStartWaitTime	Waiting time
int32	uiAutoSpoofingTime	Spoofing time

4.7 Enumerations

The following enumerations are used by the T.38 FAX Agent.

Table 3 Enumerations

Name	Description
e_MT_T38_CapType	This enum describes the protocol capabilities of the T.38 Stack.

4.7.1 e_MT_T38_CapType

Description

The e_MT_T38_CapType enum describes the protocol capabilities of the T.38 protocol stack.

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 capabilities
MT_T38_CapType_UDP	1 _D	UDP capabilities

5 T.38 FAX-Relay Features

The following chapter gives an overview of the supported T.38 FAX Relay features and the used standards for FAX Modem and FAX Tones, supported by Infineon Technologies T.38 FAX Relay Package.

5.1 Supported FAX Modem Standards

The FAX Modem is compliant to the following supported FAX Modems Standards.

Table 4 FAX Modem Standard Table

Standard	Description
V.21	V.21 Modulation is used in half-duplex mode for G3FE negotiation and control procedures.
V.27ter/2400	ITU-T recommendation for 2400/4800 bit/s modems used on the public switched network.
V.27ter/4800	ITU-T recommendation for 2400/4800 bit/s modems used on the public switched network.
V.29/7200	ITU-T recommendation for 9600 bit/s modems with a fall back rate of 7200 bit/s.
V.29/9600	ITU-T recommendation for 9600 bit/s modems with a fall back rate of 7200 bit/s.
V.17/7200	ITU-T recommendation for simplex modulation techniques used in extended G3FE applications. Up to 144000 bit/s with fall back to 12000bps, 9600bps and 7200bps.
V.17/9600	ITU-T recommendation for simplex modulation techniques used in extended G3FE applications. Up to 144000 bit/s with fall back to 12000bps, 9600bps and 7200bps.
V.17/12000	ITU-T recommendation for simplex modulation techniques used in extended G3FE applications. Up to 144000 bit/s with fall back to 12000bps, 9600bps and 7200bps.
V.17/144000	ITU-T recommendation for simplex modulation techniques used in extended G3FE applications. Up to 144000 bit/s with fall back to 12000bps, 9600bps and 7200bps.

5.2 FAX Tone Support

The following FAX tones are supported for FAX and listed in [Table 5](#).

Table 5 FAX Tone Support Table

FAX Tones	Description
CNG	Generation and detection of CNG.
CED	Generation and detection of CED.
DIS Preamble	Detection of V.21 Preamble.
ANS	Answering Tone
/ANS	Answering Tone Phase Reversal
ANSam	Answering Tone AM
./ANSsam	Answering Tone AM Phse Reversal

5.3 Supported FAX Standards

The T.38 FAX Relay Package is compliant to the following FAX Standards listed in the following tables:

Table 6 Supported FAX Standard Table

Standard	Description
T.4 including ECM	ITU-T recommendation for the coding techniques with optional error control mode ECM for T.30 The ECM support covers the coding schemes which have to be used, according to T.30, together with the ECM.
T.30	ITU-T recommendation for facsimile transmissions on PSTN.
T.38	ITU-T recommendation for G3FE transmission over IP Networks.

5.4 T.38 Protocol Stack Overview

The following list gives an overview of the supported and not supported features for the current T.38 Protocol stack implementation.

- **T.38 ITU-T Recommendation Version 0**
ASN.1 1998 syntax notation - Initial publication from (1998), Amendment (1990), Amendment 2 (02/00).
- **T.38 Annex A - ASN.1 2002 notation**
ASN.1 2002 syntax notation - Updated Recommendation (2002).
- **Signaling Protocol Support**
 - T.38 Annex B - used for H323 call establishment procedures
 - T.38 Annex D - used for SIP/SDP call establishment procedures
 - T.38 Annex E - used for H.248.1 call establishment procedures (MG/MGCP)

The T.38 protocol interface supports all necessary items for T.38 connection negotiation and establishment which are required by the signaling protocols SIP, H.323 and MG/MGC.
- **Data Rate Management Method 1**
The Data Rate Management Method 1 generates the TCF training signal locally and based on both results from the PSTN connection the data rate is determined for the high-speed connection.
- **Data Rate Management Method 2**
For Data Rate Management Method 2 the speed selection is done by the G3FE in the same way as on a regular PSTN connection.
- **IFP Packet layer support**
 - TCP Transport Protocol, T.38 Version 0 does not include the TPKT headers.
 - UDP Transport Protocol, the IFP data packets are encapsulated in UDPTL headers. The RTP protocol is not supported for UDP.
 - Maximum UDPTL buffer size is of 4096 bytes and the maximum datagram size is of 512 bytes for UDPTL.
- **UDPTL Error Correction Support**
 - T.38 Annex C - UDPTL Forward Error Correction (FEC) Scheme - The idea behind FEC is to apply the XOR operation to a group of several packets (P1 XOR P2 XOR P3 XOR P4) and transmit the result as a separate FEC packet. If there is any single packet loss in the group the packet can be restored by applying the XOR operation with the extra FEC packet. This will then restore the content of the lost packet. For example if P2 is lost the content can be get back by performing P1 XOR P3 XOR P4 XOR FEC. The advantage of the FEC error correction scheme is that it may reduce the network load using certain set of relevant parameters (uiUdpHighRateErrRecoveryPackets, uiUdpLowRateErrRecoveryPackets, uiUdpPriorPacketsForFEC).
 - UDPTL Redundancy Error Correction Scheme according to T.38 ITU-T recommendation - For redundancy N copies of each outgoing packet are sent. This increases the traffic N+1 times, but the receiver needs only

a single non-corrupted copy of the packet to restore it. For example if N=4 then 3 of 4 outgoing packets can be lost and the receiver still gets a non corrupted packet.

- The number of redundancy packets for V.21 and page data can be configured independently from each other.
- **Non standard facilities handling**
 - NFS - The Non Standard Facility (NSF) signal is used to inform the calling device about extra features which can be used during the FAX transmission (Country, Manufacture and FAX Model information).
 - Discard non standard frames to prevent non standard protocol handling.
 - Patching contents of NFS frame for example by replacing the country information to prevent non standard handshaking procedures.

5.5 Additional T.38 Features

Additional features of the T.38 Protocol stack:

- **Multi channel support**

The T.38 Protocol stack support multi channel usage. The number of channels depends on the number of available FAX Data Pump channels.

 - Each FAX Channel requires 15.5 Kilobyte.
 - If the T.38 trace feature is enabled the channel requires additional 2MB.
- **Configurable T.38 Stack Parameters**
 - IP protocol selection: TCP or UDP.
 - Rate Management Method selection.
 - Configuration of number of used packets for Redundancy and FEC.
 - Programmable FAX Data Packet size of 5, 10 and 20 milliseconds.
 - Program able output signal level.
 - Enable/Disable of spoofing feature.
 - ASN.1 configuration for Version 0 or Version 2.
 - Buffering of V.21, ECM and non ECM page data before start of modulation.

5.6 QoS Support

The following feature are supported to handle the network impairments:

- **Spoofing**

For non ECM Pages - the Spoofing mechanism inserts fill bits before end of line (EOL) to get a continuous data flow if the data is delayed by the network to prevent breaks in the modulation flow. For ECMPage - the Spoofing mechanism adds additional flags between the T.30 frames.
- **Round Trip Delay**

Round trip delays up to 7 second are supported. In case of round trip delays greater than 7 seconds, T.38 gives an time out and stops the T.38 session.

5.7 Supported T.38 Statistics

The following T.38 statistics are supported by the T.38 Protocol Stack:

- **Special Information**
 - FEC mode used by remote side
 - Redundancy mode used by remote side
 - T.30 transmission in ECM mode or non ECM mode.
 - Transaction completed - the T.30 session between facsimile devices, connected to gateways, is finished with the DCN frame.
- **Modem Standard**
 - Used modem rates and standard during the T.38 session (V.27, V.29, V.17)
- **Packet Recovery**

- Recovered packets during a T.38 session and common number of lost packets during the T.38 session.
- **Packet Loss**
 - Information about the number of consecutive lost packets.
- **Training Check**
 - Number of retrains. During the training procedure the sending facsimile sends a frame or a chain of frames and after 75 ms the TCF signal. The receiving facsimile answers with the CFR frame (training OK) or FFT signal (training not OK). The sending facsimile repeat the procedure with a lower bit rate.
- **Page Transfer**
 - Number of successful transferred pages.
- **Error Statistics**
 - Information about the number of broken lines for non ECM transferred pages.
 - Information about the number of incorrect modulated V.21 frames.
 - Information about the number of invalid ECM modulated frames.
- **T.38 Release**
 - Information about the T.38 released version.

5.8 Unsupported T.38 Features

The following features are not part of the T.38 Protocol stack implementation:

- **Transcoding Mechanism**
 - MMR and JBIG Transcoding - Optional feature for bandwidth reduction is not supported.
 - JBIG
- **IFP Packet Layer Support**
 - TPTK for TCP is not part of T.38 Version 0.
 - For UDP network transmission RTP is not supported.
- **Fill Bit Removal**
 - The optional capability to remove Fill Bits to reduce the bandwidth is not supported.
- **Signaling Protocol Support**
 - T.38 Annex G - used for H.245 Capability Definition for T.38 over RTP is not supported.

6 Source Code Organization

The source code organization of each module is illustrated in the following figures. How to compile the T.38 Protocol Stack is described in the “Programmer’s Manual for T.38 Protocol Stack” document.

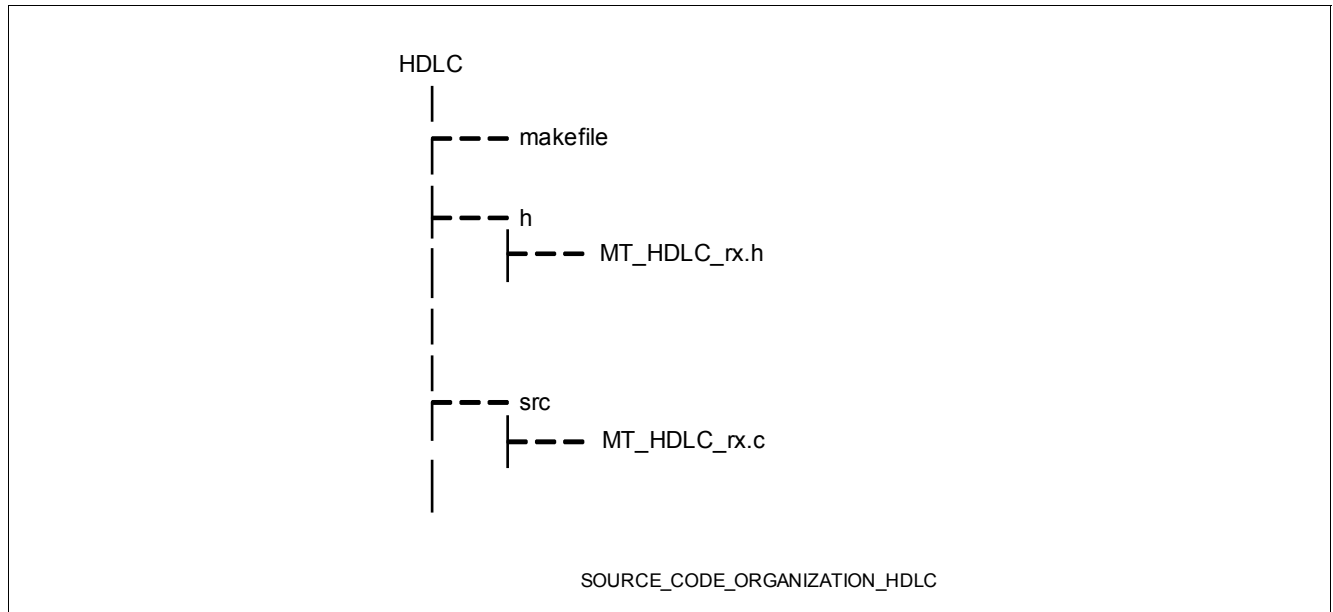


Figure 5 Source Code Organization of HDLC

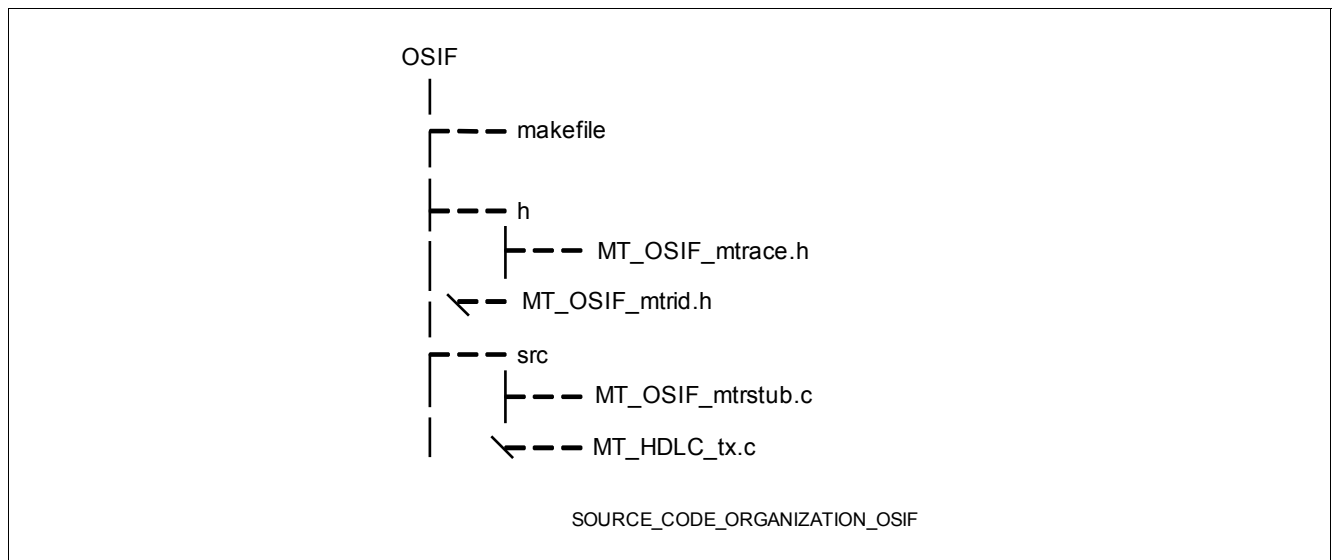


Figure 6 Source Code Organization of OSIF

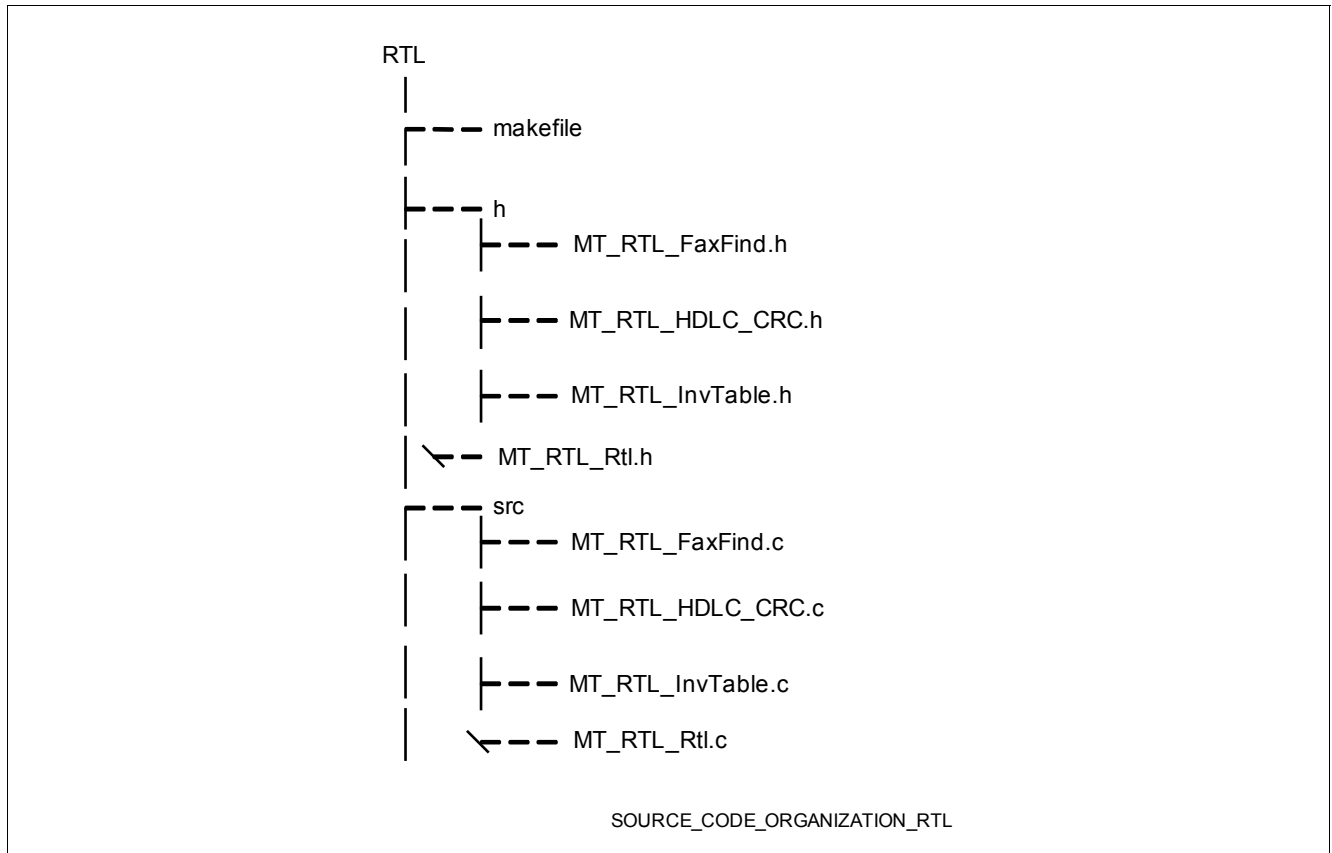


Figure 7 Source Code Organization of RTL

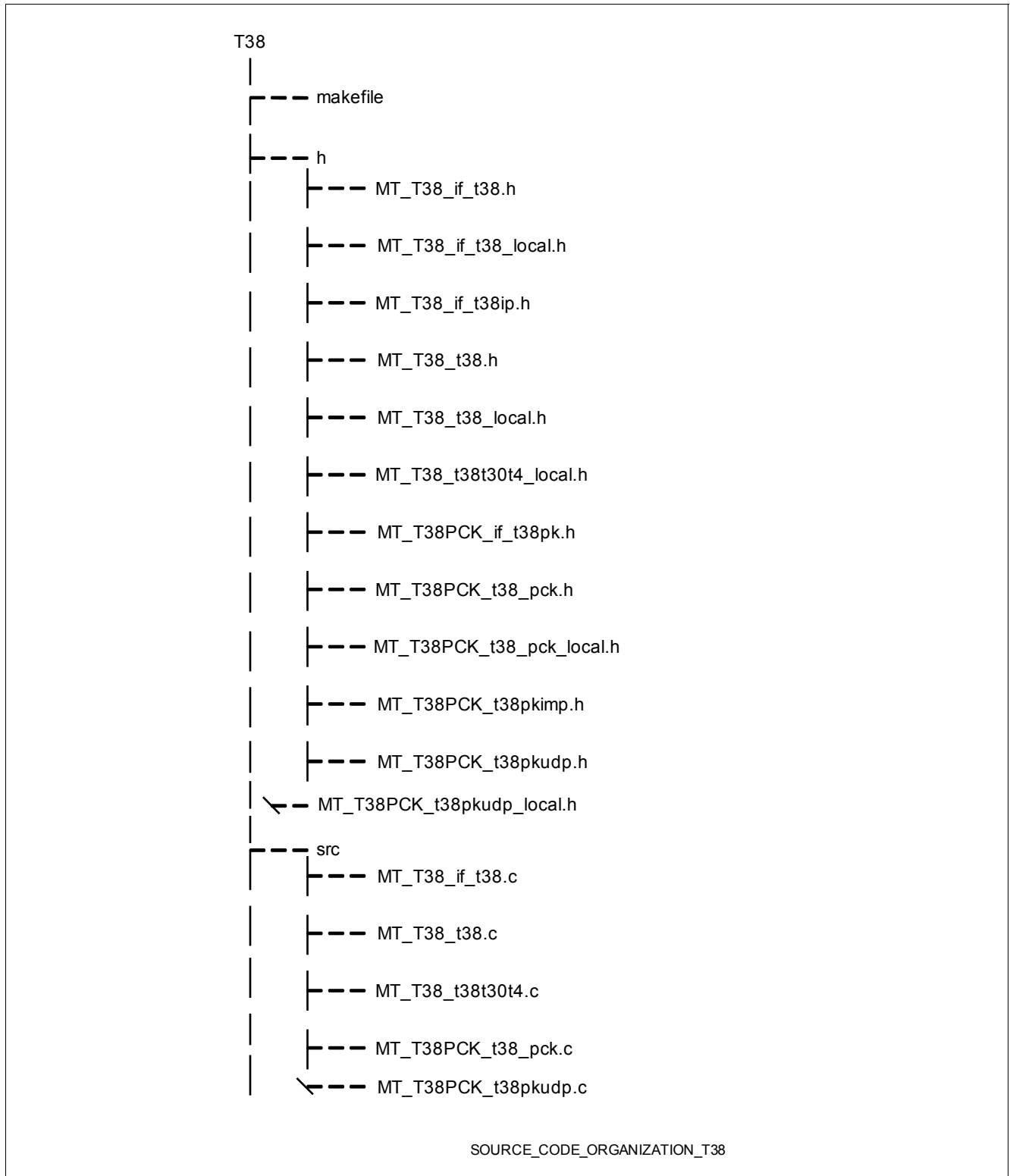


Figure 8 Source Code Organization of T.38

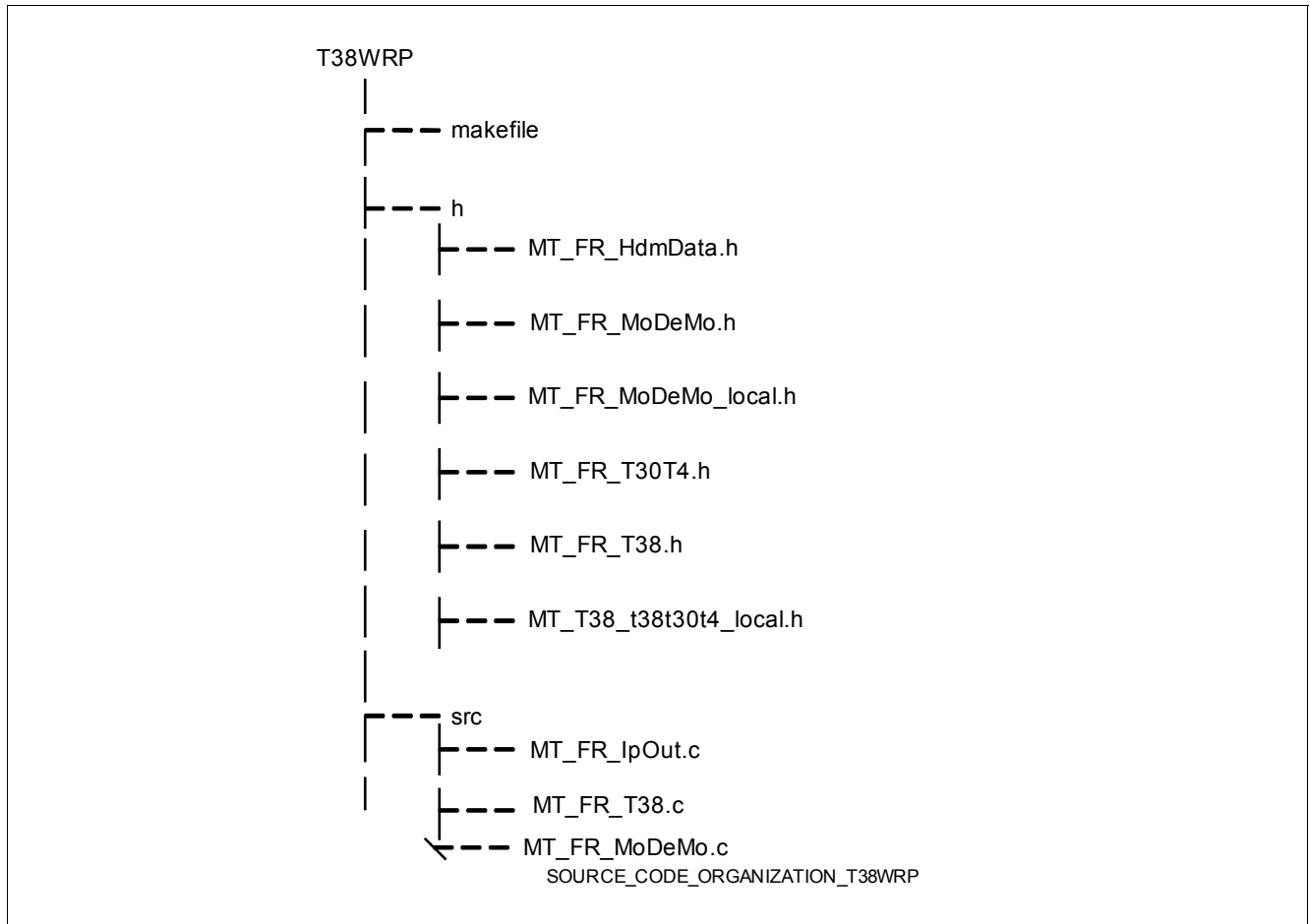


Figure 9 Source Code Organization of T38WRP

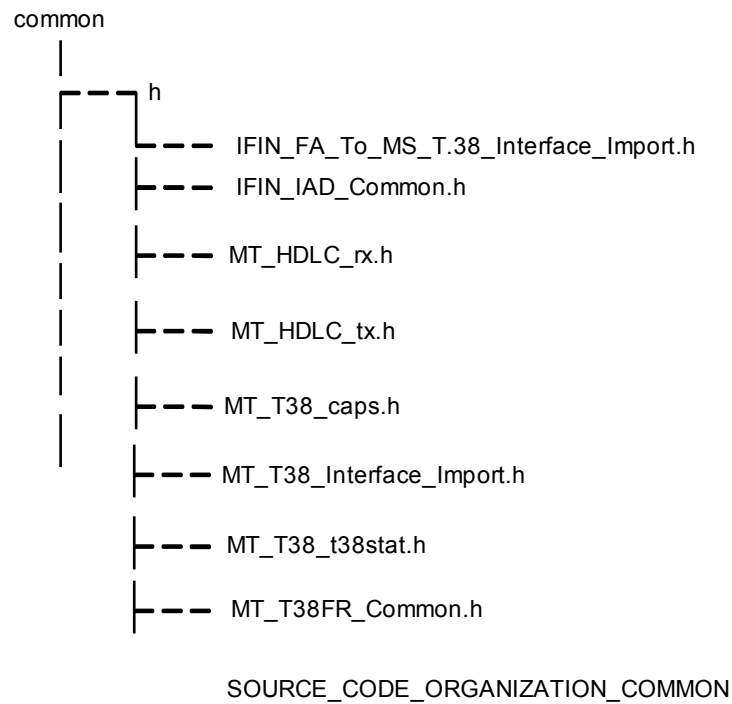


Figure 10 Source Code Organization of COMMON

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 Fax Agent Release 1.1 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