Code Affaire : **1-12-DAS-2102** N° Marché ou N° Cde : **48T03478**

Classification : **Diffusion Restreinte**

Diffusion interne : **F. BAUD** (Diffuse aux personnes concernées au sein de sa structure)

Diffusion externe : (Diffuse aux personnes concernées au sein de sa structure)

Rédigé par **T. TOURRES**	Vérifié par **J. GENDRON**	Approuvé par **F. BAUD**	Approuvé par **D. NAJAH**
Visa P/O 	Visa Jer Signé par :	Visa 	Visa

Nombre de pages : 25

Dossier d'Architecture Logicielle

du

Convertisseur ARINC 429/Ethernet

ÉVOLUTIONS

IND.	DATE	SECR.	REDACTEUR	OBJET
00	**04/03/21**		**T. Tourrès**	**Version Initiale**
01	**15/02/23**		**T. Tourrès**	**Evolutions pour Logiciel CAE V01**

SOMMAIRE

1. IDENTIFICATION

1.1. IDENTIFICATION DU LOGICIEL

Ce document constitue le Dossier d'Architecture Logicielle du convertisseur ARINC 429/Ethernet.
Il s'inscrit dans le prolongement de la STBL du logiciel ([R1]).
Le tableau suivant identifie le logiciel :

Désignation du logiciel	Référence	Fiche de Configuration Logiciel
Logiciel Applicatif CAE	DP076570LOG023 vYY(*)	DP076570FCL025
Binaire QSPI (FSBL_SSBL)	DP076570LOG001 vYY(*)	DP076570FCL004
Binaire EMMC	DP076570LOG015 vYY(*)	DP076570FCL019
Logiciel embarqué CAE	N/A	DP076570FCL026

(*): YY représente l'indice d'évolution du logiciel.

1.2. IDENTIFICATION DES DOCUMENTS DE REFERENCE

Rep.	Intitulé	Référence	Indice	Date
[R1]	STBL du CAE	DP076570SBL001	03	15/02/2023
[R2]	STI Ethernet du CAE	DP076570STI007	03	15/02/2023
[R3]	DAU du CAE	DP076570MUL001	02	08/02/2023

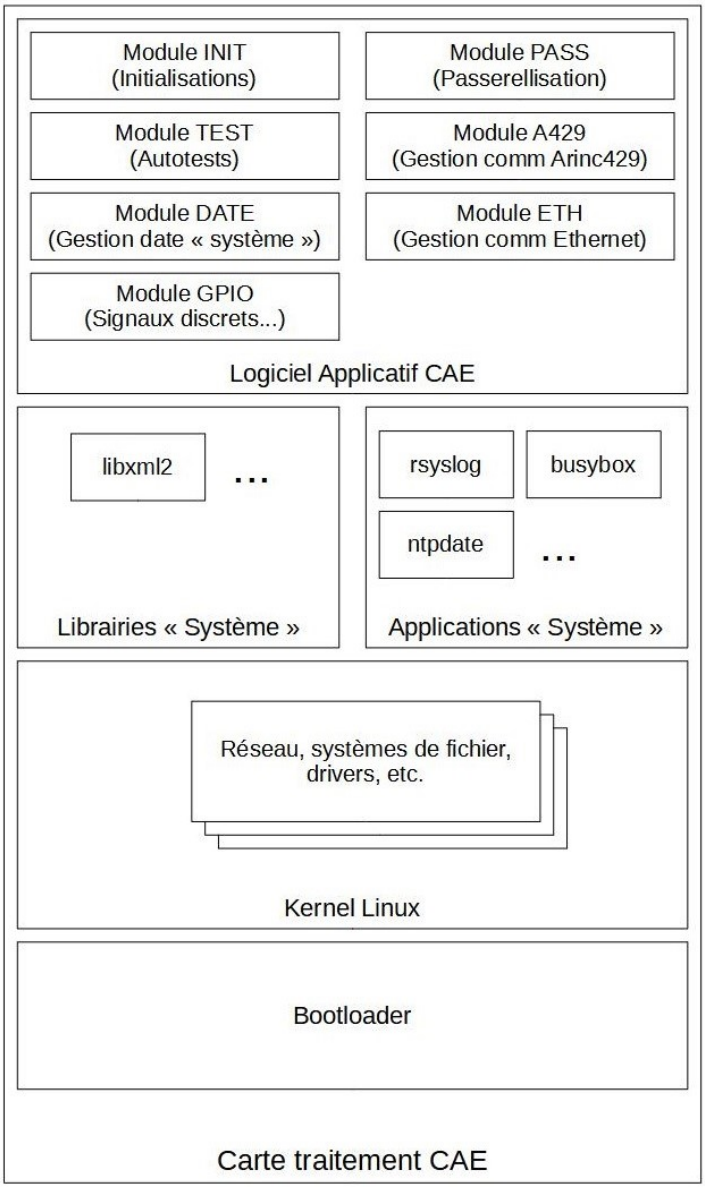
2. GLOSSAIRE ET ABREVIATIONS

Terme	Définition
A429	ARINC 429
ACU	A Confirmer Ultérieurement
ADU	A Définir Ultérieurement
ARP	Address Resolution Protocol
BF	Bon Fonctionnement
CAE	Convertisseur A429/Ethernet
CBIT	Continuous Built-In Test
CEM	Compatibilité électromagnétique
COTS	Commercial off-the-shelf
DA	Dassault Aviation
DAL	Dossier d'Architecture Logicielle
DAL-D	Development Assurance Level D
DAU	Document d'Administration et d'Utilisation
DSCP	Differentiated Services Codepoint
DTM	Dossier de Testabilité et de Maintenance
FSBL	First Stage BootLoader
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol

Terme	Définition
LLC	Logical Link Control
MAC	Media Access Control
MCS	Maintien en Conditions de Sécurité
MDI	Medium Dependent Interface
MTU	Maximum Transmission Unit
NTP	Network Time Protocol
OS	Operating System (Système d'Exploitation)
PBIT	Power-up Built-In Test
PCP	Priority Code Point
RFC	Requests For Comments
RGE	Référenciel Général Ethernet (Ethernet General Specifications)
RGS	Référenciel Général de Sécurité
ROM	Read Only Memory
RCP	Remote Copy Protocol
SS	Sous Système
SSBL	Second Stage BootLoader
SSI	Sécurité des Systèmes d'Information
SSL	Secure Sockets Layer
STB	Spécification Technique de Besoin
STBL	Spécification Technique de Besoin Logiciel
STI	Spécification Technique d'Interface
TF-A	Trusted Firmware ARM
TLS	Transport Layer Security
UAL	Unité Arithmétique et Logique
URL	Unité Remplaçable en Ligne
VLAN	Virtual Local Area Network
XML	eXtensible Markup Language

3. STRUCTURE STATIQUE

La figure suivante présente les différents modules logiciels mis en œuvre au sein du CAE :



L'intégralité du logiciel du CAE est compilé en 32 bits.
Le Bootloader représente toute la chaine de démarrage entre la mise sous tension de l'équipement jusqu'à l'exécution de l'OS. Cette chaine de démarrage inclut :

- ROM code : programmé au sein du CPU lors de sa production, il est propriétaire et non modifiable. Il prend la main à la mise sous tension, vérifie et charge le FSBL avant de lui donner la main
- FSBL : programmé dans la flash QSPI, TF-A vérifie et charge le SSBL avant de lui donner la main
- SSBL : programmé dans la flash QSPI, U-Boot vérifie et charge l'OS avant de lui donner la main

L'OS utilisé pour le logiciel du CAE est Linux, programmé dans la flash eMMC, Il comprend l'ensemble des drivers et services nécessaires au fonctionnement du réseau, des systèmes de fichier, etc…
Un certain nombre de libraires et d'applications « Système » sont également présentes, notamment :

- rsyslog : utilitaire de logs du système et des applications
- busybox : fournit différents utilitaires nécessaires au fonctionnement de Linux (shell, commandes systèmes, …) pour un environnement embarqué (un seul exécutable pour un gain de place)

- spidev : driver d'accès au bus SPI
- ntpdate : utilitaire système de mise à jour de la date et de l'heure
- libxml2 : librairie permettant la manipulation de fichiers au format XML

La liste des comptes utilisateur de l'OS et leurs privilèges, les règles de robustesse des mots de passe ainsi que les procédures de mise à jour de la configuration de messagerie sont décrites dans le document d'administration et d'utilisation de l'équipement [R3].

Mis à part le logiciel applicatif CAE, tous les composants sont open source. Tous les composants sont recompilés lors de la génération des différentes images à flasher sur le convertisseur. Une liste exhaustive des composants logiciels, incluant leurs noms et l'origine des sources, est décrite dans l'annexe 1.

Lors de la phase de développement, une procédure de veille fonctionnelle et sécuritaire est mise en place afin de ne pas intégrer de composant obsolète et de prendre en compte les patchs sécuritaires existants. Cette veille inclue la détection de failles sécuritaires éventuellement découvertes sur ces COTS mais pas encore patchées afin d'identifier des actions correctives à intégrer au développement.

4. SPECIFICATIONS DES COMPOSANTS ET MODULES

4.1. BOOTLOADERS

Le bootloader qui charge et lance l'OS Linux (SSBL) avec les paramètres appropriés est U-Boot.
U-Boot est chargé et lancé par TF-A (FSBL).
TF-A est lui-même vérifié et chargé par le ROM code, qui est le code read-only chargé par STMicroelectronics dans le CPU (STM32MP151C), et qui ne peux pas être modifié mais est configuré au travers de registres de mémoire OTP.
Tous ces éléments mettent en œuvre la chaine de secureboot (cf. §9) et sont inclus dans ce module pour concrètement désigner toute la chaine de démarrage jusqu'à l'OS Linux.

4.2. OS LINUX

Le logiciel applicatif du CAE se repose sur l'utilisation du noyau Linux auquel est ajouté un certain nombre de librairies et d'applications « système » qui constituent le rootfs, mais réduit au strict minimum. Les librairies et applications « Système » embarquées seront listées dans le DAL et leur présence et usage seront justifiés.

De la même façon, les services de l'OS seront présents et activés uniquement si leur utilisation est justifiée, les autres seront absents ou désactivés. Les services présents et activés seront configurés en accord avec les exigences et recommandations de sécurité applicables. En particulier, les services d'accès à distance (SSH…) seront absents sur les équipements de série, limitant l'accès au shell à une liaison série inaccessible depuis l'extérieur du boîtier.

La mémoire de masse (eMMC) utilisées par l'OS du CAE est partitionnée ainsi :

- une 1ère partition, montée en RO (Read Only), contenant l'ensemble des fichiers images (binaires) signés
- une 2ème partition, montée en RO (Read Only), contenant les fichiers de configuration de l'OS
- une 3ème partition permettant le stockage de données pour des besoins de debug

Le rootfs est contenu dans un fichier binaire de type SquashFS monté en loopback depuis le fichier image signé. Il est donc par définition en lecture seule. L'arborescence du rootfs est donc figée et non modifiable. Elle est également réduite au strict nécessaire imposé par les librairies et applications « Système » embarquées, dont la description sera donnée dans le DAL.

Le CAE a pour vocation de fonctionner en standalone et le système Linux n'est présent que pour servir de base aux fonctionnalités applicatives du CAE.

Logiciel applicatif

Le logiciel applicatif installé dans le filesystem met en œuvre les fonctionnalités du CAE et est architecturé en modules décrits ci-dessous.

4.2.1. MODULE INIT

Ce module constitue le point d'entrée du logiciel applicatif.
Il réalise le début de la phase de « pré-initialisation » :

- Initialisation du module GPIO (pour la lecture du pin programming)
- Lecture et vérification des fichiers de configuration du CAE (cf. §8.1) et de passerellisation (cf. §8.2)
- Application de la configuration réseau
- Démarrage des autres modules du logiciel

4.2.2. MODULE GPIO

Ce module gère les entrées/sorties discrètes, notamment le filtrage des entrées et le réarmement des sorties, ainsi que les I/O génériques (signaux de powergood, interruptions, LEDs…).
Il fournit les services de lecture et écriture de ces I/O aux autres modules.

4.2.3. MODULE ETH

Ce module permet l'émission et réception de messages sur l'interface Ethernet : messages d'états et messages passerellisés/à passerelliser. Il s'appuie sur le module TEST pour récupérer les résultats d'autotests et il est utilisé par le module PASS en mode opérationnel.

4.2.4. MODULE TEST

Au démarrage, ce module finalise la phase de « pré-initialisation », c'est-à-dire qu'il effectue les tests de bon fonctionnement du CPU et de l'interface Ethernet afin que le premier message d'état « initialisation en cours » puisse être émis.
Le module réalise ensuite les tests de démarrage (PBIT).
En mode opérationnel, ce module est en charge des tests réalisés en continu (CBIT).
Il s'appuie sur les modules :

- DATE pour récupérer l'état de synchronisation de la date avec les serveurs NTP
- A429 pour récupérer l'état des tests du bus A429 (rebouclage…)
- GPIO pour récupérer l'état des tests de cohérence des E/S discrètes
- INIT et PASS pour récupérer l'état des tests sur les fichiers de configuration

Lors de la détection de panne, un filtrage des pannes fugitives est effectué pour le mot d'état, par comptage pour faire remonter les bagottements de pannes continues (+2 à l'incrément, -1 au décrément, avec un seuil de 6 pour la déclaration d'une panne, valeurs configurables au sein du fichier XML décrit au §8.1 Fichier de configuration du logiciel Applicatif CAE)

4.2.5. MODULE DATE

Ce module réalise la gestion de la date système du CAE.
Lorsque le module DATE est démarré par le module INIT, celui-ci lui indique l'adresse Ethernet du serveur primaire et secondaire.
Le mécanisme de synchronisation NTP est décrit dans le document [R2], le principe étant de mettre en œuvre le client NTP de l'OS (ntpdate) pour envoyer des requêtes périodiquement aux deux serveurs NTP et de

récupérer le résultat de ces requêtes pour en déduire la validité de la synchronisation NTP. L'heure de l'OS Linux est automatiquement synchronisée par ntpdate si un des 2 serveurs répond.
Il doit fournir au module TEST le statut (synchronisé ou non) de la synchronisation avec les serveurs NTP : OK si synchronisation avec un des deux serveurs.

4.2.6. MODULE A429

Ce module est un utilitaire utilisé par le module PASS pour émettre/recevoir des messages sur le bus A429.
Il est aussi responsable du test continu de cette interface et de la remontée du résultat au module TEST.

4.2.7. MODULE PASS

Au démarrage, ce module vérifie les valeurs des fichiers de configuration applicables puis les utilise pour son initialisation.
Une fois l'équipement en mode opérationnel, ce module assure la mise en œuvre de la fonctionnalité de passerellisation des messages entre les interfaces Ethernet et A429, suivant les règles définies dans les fichiers de configuration de messagerie.
Il repose sur le module A429 pour émettre et recevoir les messages sur les bus A429 et sur le module ETH pour émettre et recevoir les messages sur Ethernet.

5. ALLOCATIONS FONCTIONNELLES

Le tableau suivant présente la répartition des exigences issues de la STBL (document [R1]) entre les différents composants et modules du logiciel du CAE :

Exigence	chaine de démarrage (CPU ROM code, bootloader, OS, APP)	OS Linux (et ses services, sa configuration)	Modules applicatifs							remarque / justification
			INIT	GPIO	TEST	DATE	ETH	A429	PASS	
AVSIMAR.SW.CAE.T0010					X					
AVSIMAR.SW.CAE.T0020					X		X			
AVSIMAR.SW.CAE.T0030							X			
AVSIMAR.SW.CAE.T0040							X			
AVSIMAR.SW.CAE.T0050							X			
AVSIMAR.SW.CAE.T0060							X			
AVSIMAR.SW.CAE.T0070							X			
AVSIMAR.SW.CAE.T0080						X				
AVSIMAR.SW.CAE.T0090						X				
AVSIMAR.SW.CAE.T0100			X			X				
AVSIMAR.SW.CAE.T0101			X			X				
AVSIMAR.SW.CAE.T0102						X				
AVSIMAR.SW.CAE.T0105				X						
AVSIMAR.SW.CAE.T0110					X					
AVSIMAR.SW.CAE.T0120			X	X					X	
AVSIMAR.SW.CAE.T0130			X		X					
AVSIMAR.SW.CAE.T0140			X		X					
AVSIMAR.SW.CAE.T0141				X						
AVSIMAR.SW.CAE.T0150					X					

Exigence	chaine de démarrage (CPU ROM code, bootloader, OS, APP)	OS Linux (et ses services, sa configuration)	Modules applicatifs							remarque / justification
			INIT	GPIO	TEST	DATE	ETH	A429	PASS	
AVSIMAR.SW.CAE.T0160									X	
AVSIMAR.SW.CAE.T0170									X	
AVSIMAR.SW.CAE.T0180									X	
AVSIMAR.SW.CAE.T0181									X	
AVSIMAR.SW.CAE.T0182									X	
AVSIMAR.SW.CAE.T0190									X	
AVSIMAR.SW.CAE.T0200									X	
AVSIMAR.SW.CAE.T0210		X								service rsyslog
AVSIMAR.SW.CAE.T0220		X								
AVSIMAR.SW.CAE.T0230		X								
AVSIMAR.SW.CAE.T0240					X					
AVSIMAR.SW.CAE.T0250					X					
AVSIMAR.SW.CAE.T0260					X		X			
AVSIMAR.SW.CAE.T0270				X	X					
AVSIMAR.SW.CAE.T0290		X			X					
AVSIMAR.SW.CAE.T0300					X		X			
AVSIMAR.SW.CAE.T0310				X	X					
AVSIMAR.SW.CAE.T0320					X					
AVSIMAR.SW.CAE.T0321					X					
AVSIMAR.SW.CAE.T0322					X					
AVSIMAR.SW.CAE.T0330					X					
AVSIMAR.SW.CAE.T0340					X					
AVSIMAR.SW.CAE.T0350					X					
AVSIMAR.SW.CAE.T0371				X	X					
AVSIMAR.SW.CAE.T0380					X					

Exigence	chaine de démarrage (CPU ROM code, bootloader, OS, APP)	OS Linux (et ses services, sa configuration)	Modules applicatifs							remarque / justification
			INIT	GPIO	TEST	DATE	ETH	A429	PASS	
AVSIMAR.SW.CAE.T0381				X	X					
AVSIMAR.SW.CAE.T0382					X					
AVSIMAR.SW.CAE.T0390					X					+ gestion documentaire
AVSIMAR.SW.CAE.T0421					X				X	
AVSIMAR.SW.CAE.T0430					X					
AVSIMAR.SW.CAE.T0441				X	X					
AVSIMAR.SW.CAE.T0442					X			X		
AVSIMAR.SW.CAE.T0443					X			X		
AVSIMAR.SW.CAE.T0444					X		X			
AVSIMAR.SW.CAE.T0450		X	X							
AVSIMAR.SW.CAE.T0451		X								Device tree
AVSIMAR.SW.CAE.T0460			X				X			
AVSIMAR.SW.CAE.T0470			X							
AVSIMAR.SW.CAE.T0480		X	X							
AVSIMAR.SW.CAE.T0490										pris en charge par la gestion documentaire (STI §6)
AVSIMAR.SW.CAE.T0500				X						
AVSIMAR.SW.CAE.T0511				X						
AVSIMAR.SW.CAE.T0512		X		X						
AVSIMAR.SW.CAE.T0513		X		X						
AVSIMAR.SW.CAE.T0514			X							
AVSIMAR.SW.CAE.T0515					X					
AVSIMAR.SW.CAE.T0520		X								
AVSIMAR.SW.CAE.T0530		X								+ mise en œuvre du filesystem OS (montage en lecture seule)
AVSIMAR.SW.CAE.T0540	X									
AVSIMAR.SW.CAE.T0550	X									

Exigence	chaine de démarrage (CPU ROM code, bootloader, OS, APP)	OS Linux (et ses services, sa configuration)	Modules applicatifs							remarque / justification
			INIT	GPIO	TEST	DATE	ETH	A429	PASS	
AVSIMAR.SW.CAE.T0560	X	X								
AVSIMAR.SW.CAE.T0570	X	X								
AVSIMAR.SW.CAE.T0590		X								
AVSIMAR.SW.CAE.T0600		X	X						X	
AVSIMAR.SW.CAE.T0610		X								
AVSIMAR.SW.CAE.T0620		X								
AVSIMAR.SW.CAE.T0640		X								
AVSIMAR.SW.CAE.T0641	X									
AVSIMAR.SW.CAE.T0650	X									
AVSIMAR.SW.CAE.T0660	X									
AVSIMAR.SW.CAE.T0670	X									
AVSIMAR.SW.CAE.T0680		X								+ gestion documentaire
AVSIMAR.SW.CAE.T0690		X								
AVSIMAR.SW.CAE.T0700		X								
AVSIMAR.SW.CAE.T0710		X								
AVSIMAR.SW.CAE.T0720										prise en charge par l'équipe projet
AVSIMAR.SW.CAE.T0730										prise en charge par l'environnement de développement
AVSIMAR.SW.CAE.T0750		X								
AVSIMAR.SW.CAE.T0760	X	X								+ procédé de production (lien entre environnement de dev et chargement du logiciel en usine)
AVSIMAR.SW.CAE.T0770		X								
AVSIMAR.SW.CAE.T0780		X								
AVSIMAR.SW.CAE.T0790		X								service rsyslog
AVSIMAR.SW.CAE.T0800										prise en charge par le design matériel
AVSIMAR.SW.CAE.T0810	X									

Exigence	chaine de démarrage (CPU ROM code, bootloader, OS, APP)	OS Linux (et ses services, sa configuration)	Modules applicatifs							remarque / justification
			INIT	GPIO	TEST	DATE	ETH	A429	PASS	
AVSIMAR.SW.CAE.T0820										prise en charge par l'architecture logicielle
AVSIMAR.SW.CAE.T0830		X								
AVSIMAR.SW.CAE.T0840		X								
AVSIMAR.SW.CAE.T0850										prise en charge par la gestion documentaire
AVSIMAR.SW.CAE.T0860										prise en charge par la gestion documentaire
AVSIMAR.SW.CAE.T0870										prise en charge par le procédé de production
AVSIMAR.SW.CAE.T0880										prise en charge par le procédé de livraison
AVSIMAR.SW.CAE.T0890										prise en charge par les échanges entre l'équipe projet et DA
AVSIMAR.SW.CAE.T0900										prise en charge par les échanges entre l'équipe projet et DA
AVSIMAR.SW.CAE.T0910		X								
AVSIMAR.SW.CAE.T0920		X								
AVSIMAR.SW.CAE.T0930		X								
AVSIMAR.SW.CAE.T0940		X								
AVSIMAR.SW.CAE.T0950										prise en charge par l'environnement de développement
AVSIMAR.SW.CAE.T0960										prise en charge par l'architecture logicielle et le procédé de livraison (antivirus)
AVSIMAR.SW.CAE.T0970										prise en charge par l'architecture logicielle
AVSIMAR.SW.CAE.T0980										prise en charge par le procédé de livraison
AVSIMAR.SW.CAE.T0990										prise en charge par l'architecture logicielle
AVSIMAR.SW.CAE.T1000										détails dans les documents de conception
AVSIMAR.SW.CAE.T1010										prise en charge par l'architecture logicielle
AVSIMAR.SW.CAE.T1020										prise en charge par l'architecture logicielle
AVSIMAR.SW.CAE.T1030										prise en charge par l'architecture logicielle
AVSIMAR.SW.CAE.T1040										prise en charge par l'environnement de développement et la gestion documentaire

Exigence	chaine de démarrage (CPU ROM code, bootloader, OS, APP)	OS Linux (et ses services, sa configuration)	Modules applicatifs							remarque / justification
			INIT	GPIO	TEST	DATE	ETH	A429	PASS	
AVSIMAR.SW.CAE.T1050	X	X								
AVSIMAR.SW.CAE.T1060										prise en charge par l'architecture logicielle
AVSIMAR.SW.CAE.T1071		X								

6. ALLOCATION DES RESSOURCES INFORMATIQUES

Le logiciel s'exécute sur une carte CPU fabriquée par ATOS, à base de STM32MP1.
Le STM32MP151C est un MPU de chez STMicroelectronics pourvu d'un cœur ARM A7 et d'un cœur ARM M4.

La carte CPU possède 2 mémoires de stockage :

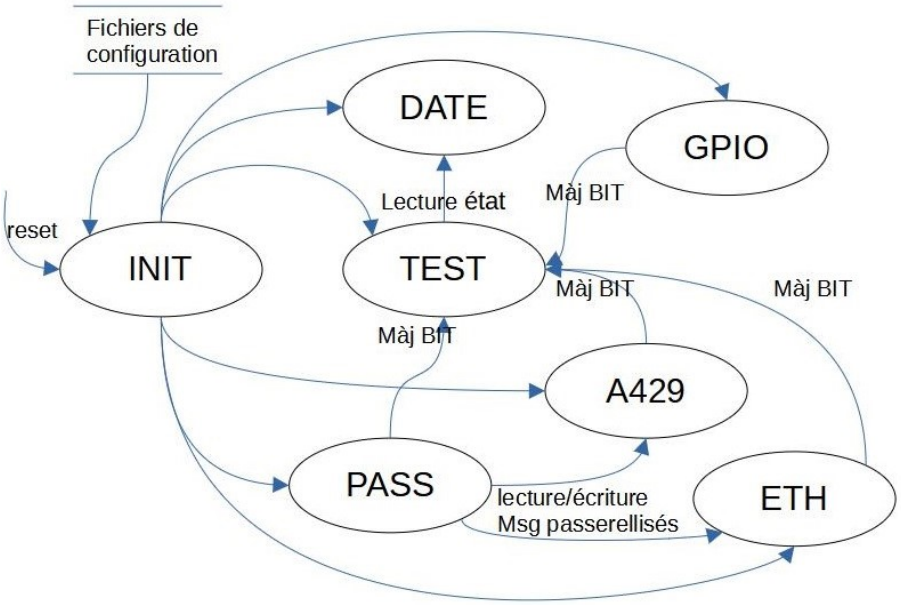
- Deux flash de 64 Mio sur une interface QSPI (logiciellement montées comme un unique périphérique)
- Une flash de 8 Go sur une interface eMMC

La flash QSPI permet de stocker le binaire de boot : FSBL+SSBL
Quant à la flash eMMC, elle contient les autres binaires (script de boot, kernel, device tree, initrd, rootfs, applicatif).
Elle est vue comme une mémoire de masse classique de type disque et elle est partitionnée ainsi :

MBR (Master Boot Record) = 512 o	Contient la table des partitions
Première partition : **Factory repository** Système de fichier : ext3 / read-only Taille : 150 Mio Label = FACTREPO_LBL	Contient les images des binaires et des filesystems : • uImage (kernel) • initrd.img • devicetree.dtb (binaire du device-tree) • rootfs.img • opt.img (logiciel applicatif) Et contient également le fichier de script du Boot Loader
Seconde partition : **Configuration système** Système de fichier : ext3 / read-only Taille : 2 Mio Label = CONFIG_LBL	Utilisée uniquement en phase de debug (contient les fichiers de configuration de l'OS et de messagerie ayant besoin d'être modifiés pendant la mise au point). N'est pas utilisée en mode opérationnel.
Troisième partition : **Data** Système de fichier : btrfs / read-write Taille : taille de la flash moins la taille des 2 1ères partitions Label = DATA_LBL	Utilisée uniquement en phase de debug (contient les données liées aux logiciels applicatifs et les fichiers de log). N'est pas utilisée en mode opérationnel.

7. ASPECTS DYNAMIQUES

La figure suivante représente les aspects dynamiques entre les différents modules du logiciel applicatif CAE :



8. DONNEES PARTAGEES

8.1. FICHIER DE CONFIGURATION DU LOGICIEL APPLICATIF CAE

Le fichier de configuration du logiciel applicatif CAE permet de configurer toutes les valeurs dynamiques à l'exception des règles de passerellisation qui sont configurées au travers des fichiers de configuration dédiés (cf. §8.2 Fichiers de configuration de la passerelisation).
C'est un fichier XML, encodé en UTF-8.
Il doit se situer à /etc/config et se nommer appSW_XYZ.xml avec :

- X, Y et Z = 0 ou 1
- X = valeur de PINPROG1
- Y = valeur de PINPROG2
- Z = valeur de PINPROG3

Il doit contenir le nœud principal « appSW-config », qui lui doit contenir obligatoirement les nœuds suivants :

- « eth » : configuration des paramètres liés à la communication du CAE sur ETH1
- « etat » : paramètres pour l'émission du message d'état
- « ident : paramètres pour l'émission du message d'identification
- « date » : configuration des paramètres liés à la datation
- « watchdog » : configuration de l'activation du watchdog
- « failure » : configuration de la gestion des défauts

8.1.1. NŒUD « ETH »

Ce nœud doit contenir obligatoirement les propriétés suivantes :

- « vitesse » : vitesse de l'interface Ethernet
  - Valeur : « 10 » ou « 100 »
- « duplex » : type de duplex
  - Valeur : « half » ou « full »
- « autoneg » : activation de l'auto-négociation
  - Valeur : « on » ou « off »
- « flowControl » : activation du contrôle de flux
  - Valeur : « on » ou « off »
- « MDI » : configuration du croisement MDI
  - Valeur : « auto », « MDI » ou « MDI-X »

8.1.2. NŒUD « ETAT »

- « id » : identifiant du Data Group du message « Etat » (cf. STI [R2])
  - Valeur : mot de 32bits représenté en hexadécimal préfixé : « 0xNNNNNNNN »
- « ipAddrs » : Adresse IP du destinataire du message « Etat » (cf. STI [R2])
  - Valeur : format IP
- « port » : numéro de port correspondant au protocole « Etat » (cf. STI [R2])
  - Valeur : en décimal
- « period » : temps en secondes entre deux envois du message « Etat » (cf. STI [R2])
  - Valeur : en décimal

8.1.3. NŒUD « IDENT »

- « id » : identifiant du Data Group du message « Identification » (cf. STI [R2])
  - Valeur : mot de 32bits représenté en hexadécimal préfixé : « 0xNNNNNNNN »
- « ipAddrs » : Adresse IP du destinataire du message « Identification » (cf. STI [R2])
  - Valeur : format IP
- « port » : numéro de port correspondant au protocole « Identification » (cf. STI [R2])
  - Valeur : en décimal
- « period » : temps en secondes entre deux envois du message « Identification » (cf. STI [R2])
  - Valeur : en décimal

8.1.4. NŒUD « DATE »

Ce nœud doit contenir obligatoirement les propriétés suivantes :

- « primaryNtpServer » : adresse IP du serveur NTP primaire (cf. STI [R2])
  - Valeur : format IP
- « secondaryNtpServer » : adresse IP du serveur NTP secondaire (cf. STI [R2])
  - Valeur : format IP
- « primarySrvErrorBeforeSecondaryUsing » : nombre d'erreur de synchronisation avec le serveur principal avant l'envoi de requêtes de synchronisation au serveur secondaire
  - Valeur : en décimal
- « syncQueryPeriod » : temps en secondes entre deux requêtes de synchronisation lorsque le CAE est synchronisé
  - Valeur : en décimal
- « notSyncQueryPeriod » : temps en secondes entre deux requêtes de synchronisation lorsque le CAE n'est pas synchronisé
  - Valeur : en décimal
- « primaryStratum » : numéro de stratum du serveur principal
  - Valeur : en décimal
- « secondaryStratum » : numéro de stratum du serveur secondaire
  - Valeur : en décimal

8.1.5. NŒUD « WATCHDOG »

Ce nœud doit contenir obligatoirement la propriété suivante :

- « activation » : activation ou inhibition du watchdog
  - Valeur : « on » ou « off »

8.1.6. NŒUD « FAILURE »

Ce nœud doit contenir obligatoirement les propriétés suivantes :

- « incStep » : pas d'incrément du compteur de défaut. Utilisé pour le filtrage des défauts, lors de la présence d'un défaut
  - Valeur : en décimal, entre 1 et 255 inclus
- « decStep » : pas de décrément du compteur de défaut. Utilisé pour le filtrage des défauts, lors de l'absence d'un défaut
  - Valeur : en décimal, entre 1 et 255 inclus
- « treshold » : seuil de déclaration d'un défaut
  - Valeur : en décimal, entre 0 et 255 inclus

8.1.7. EXEMPLE DE FICHIER DE CONFIGURATION DE CAE

Fichier /etc/config/appSW_010.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<appSW-config>
	<eth
		vitesse="100"
		duplex="full"
		autoneg="on"
		flowControl="on"
		MDI="MDI"
	/>
	<etat
		id="0x1427"
		ipAddrs="192.0.1.30""
		port="50900"
		period="1"
	/>
	<ident
		id="0x1465"
		ipAddrs="192.0.1.30""
		port="50900"
		period="10"
	/>
	<date
		primaryNtpServer="192.0.3.2"
		secondaryNtpServer="192.0.3.1"
		primarySrvErrorBeforeSecondaryUsing="10"
		syncQueryPeriod=”64”
		notSyncQueryPeriod="4"
		primaryStratum="11"
		secondaryStratum="14"
	/>
	<watchdog
		activation="on"
	/>
	<failure
		incStep="2"
		decStep="1"
		threshold="6"
	/>
</appSW-config>
```

8.2. FICHIERS DE CONFIGURATION DE LA PASSERELISATION

C'est la note technique DP076570NTE006 qui décrit le format du fichier de configuration de messagerie. Les réponses/contre-propositions de DA sont faits au travers de l'ECM 15.

9. MISE EN ŒUVRE DU SECURE BOOT

9.1. CONSTRUCTION DES IMAGES

9.1.1. IMAGE DE BOOTLOADER

L'image de bootloader représente le binaire programmé sur la mémoire flash QSPI et contenant les éléments nécessaires au démarrage sécurisé du système (OS + applicatif) programmé sur l'autre mémoire flash.
Les étapes de construction de l'image de bootloader sont :

1. Depuis les sources, recompiler TF-A avec le Makefile inclus :
   1.1. La compilation génère :
      1.1.1. Tfa.bin : l'exécutable du FSBL (BL2)
      1.1.2. Fw-config.dtb : fichier de configuration du firmware
      1.1.3. Bl32.bin : l'exécutable « secure monitor SP-MIN » (BL32), la 2ème partie de TF-A
      1.1.4. Bl32.dtb : fichier de configuration du BL32
2. Signature du BL2 avec STM32MP_SigningTool
   2.1. L'outil prend en entrée :
      2.1.1. Tfa.bin (1.1.1)
      2.1.2. Les clés publiques et privées (ECC 256bits)
      2.1.3. Les adresses mémoire de chargement, point d'entrée
   2.2. L'outil génère :
      2.2.1. Tfa-signed.bin l'exécutable signé
      2.2.2. Fichiers PKH à utiliser sur la cible pour inscrire les informations nécessaires à la vérification de la signature dans la mémoire OTP du CPU (eFuse, hash de clé publiques)
3. Depuis les sources, recompiler U-Boot (SSBL/BL33) :
   3.1. Les sources contiennent :
      3.1.1. La fonctionnalité de vérification de signature RSA4096
      3.1.2. La valeur de la clé publique permettant de vérifier les signatures des images système
   3.2. La compilation génère :
      3.2.1. U-boot-nodtb.bin : l'exécutable de U-Boot
      3.2.2. U-boot.dtb : fichier de configuration de U-Boot
4. Création de l'image signée au format FIP, avec l'outil fiptool
   4.1. L'outil prend en entrée :
      4.1.1. Bl32.bin (1.1.3) : option –tos-fw
      4.1.2. Bl32.dtb (1.1.4) : option –fw-config
      4.1.3. U-boot-notdb.bin (3.2.1) : option –nt-fw
      4.1.4. U-boot.dtb (3.2.2) : option –hw-config
      4.1.5. La clé nécessaire à la signature (ECC 256bits, identique à 2.1.2)
   4.2. L'outil génère :
      4.2.1. SSBL_FIP_signed.bin
5. Création de l'image disque respectant les contraintes de partitionnement du STM32MP :
   5.1. Partition 1 : « fsbl_1 » : Tfa-signed.bin (2.2.1)
   5.2. Partition 2 : « fsbl_2 » : Tfa-signed.bin (2.2.1)
   5.3. Partition 3 : « fip » : SSBL_FIP_signed.bin (4.2.1)
   5.4. Partition 4 : « ident » : identifiants de version logicielle AVANTIX de l'ensemble de l'image

L'image disque résultante est chargée sur la mémoire QSPI avec l'outil STM32_Programmer.
C'est ce même outil qui permet d'inscrire en mémoire OTP du CPU (BSEC) les données générées à l'étape 2.2.2 et de « fermer le système » (« close the device » = activer le secure boot en programmant définitivement l'OTP WORD 0 à « Closed state »).

9.1.2. IMAGES SYSTEME

Les images système correspondent aux composants suivants, chargés sur la mémoire EMMC :

- script de boot
- uImage (kernel)
- initrd.img
- devicetree.dtb (binaire du device-tree)
- rootfs.img
- opt.img (logiciel applicatif)

Chacune de ces images est générée en association avec :

- Un fichier de hash sha256
- Une signature RSA4096 générée avec le pendant privé de la clé présente dans U-Boot (étape 3.1.2 du chapitre précédent).
- Des informations d'identifications du composant :
  - Numéro de build (incrémenté automatiquement à chaque compilation/génération)
  - Date (idem)
  - Indice de version
  - Référence du logiciel

Ces images sont regroupées en une image, avec ses propres informations d'identification englobantes (build, date, ind, ref). Cette image est programmée dans la mémoire flash EMMC avec U-Boot.

9.2. SEQUENCE DE DEMARRAGE SECURISE

L'authentification des logiciels repose sur le mécanisme appelé « chaîne de confiance » (chain-of-trust) : le premier processus de démarrage (BL : BootLoader), se reposant sur une racine matérielle de confiance, authentifie le second, le second processus authentifie le troisième, etc. :

1. ROM CODE : à la mise sous tension le ROM CODE a la main dans la TrustedZone du CPU
   1.1. Sa configuration indique un démarrage sur QSPI en mode sécurisé
   1.2. Il utilise le hash de clé ECC 256bits des e-fuses du CPU pour vérifier la signature du BL2 (TF-A : TrustedFirmware-A)
   1.3. Il charge le BL2 en SYSRAM et lui donne la main
2. BL2 (TF-A) :
   2.1. Il utilise le hash de la clé ECC 256bits des e-fuses du CPU pour vérifier la signature de l'archive FIP (BL32+BL33)
   2.2. Il charge le BL32 (TF-A SP-MIN) en SYSRAM et le BL33 (U-Boot) en DDR
   2.3. Il donne la main au BL32
3. BL32 (TF-A SP-MIN) :
   3.1. Il donne la main au BL33
4. BL33 (U-Boot)
   4.1. Il utilise la clé publique (RSA 4096bits) intégrée à son code pour vérifier la signature du script de boot puis l'exécute
   4.2. Le script de boot utilise cette même clé pour vérifier toutes les signatures des autres composants de la flash EMMC : uImage (kernel), initrd.img, devicetree.dtb, rootfs.img, opt.img (logiciel applicatif)
   4.3. Les images étant toutes authentifiées, le démarrage se poursuit de manière standard : chargement en DDR puis lancement de l'OS Linux par la commande bootm.

Si une erreur intervient lors de cette séquence, comme une signature non vérifiée, la séquence est interrompue et le CPU est bloqué pour ne plus exécuter aucun code. La sortie discrète de bon fonctionnement reste à KO (sa valeur par défaut) et aucun message d'état n'est émis.

10. ORGANISATION

10.1. ARBORESCENCE DES FICHIERS

L'arborescence des fichiers et des répertoires sur la machine hôte (machine de développement) est la suivante :

```
Répertoire de développement racine /
|
>-- 00.ID/ : Identifiants chapeau du firmware
|
>-- 01.buildtools/ : contient les outils logiciels nécessaires à la génération des binaires
|
>-- 02.build/ : contient l'ensemble des sources compilées
| |
| >-- 01.bootldr/ : répertoire des sources du boot loader
| |
| >-- 02.kernel/ : répertoire des sources de l'OS
| |
| >-- 03.devicetree/ : répertoire des sources du device tree
| |
| >-- 04.initrd/ : répertoire des sources de l'initial ram disk
| |
| >-- 05.rootfs/ : répertoire des sources du rootfs (librairies et applications système)
| |
| >-- 06.apps/ : répertoire des sources du logiciel Applicatif
|
>-- 03.filesystems/ : contient les file systems générés (initrd, rootfs et apps)
|
>-- 04.images/ : contient les fichiers binaires images des file systems générés
|
>-- 05.flashables/ : contient les binaires à copier pour la mise à jour logicielle
```

10.2. REGLES DE NOMMAGE

Sans objet

ANNEXE 1 :

Liste des composants logiciels, leurs noms et leurs sources

arborescence		nom et version	origine des sources
00.fsbl		arm-trusted-firmware	https://github.com/STMicroelectronics/arm-trusted-firmware/tags
01.ssbl		u-boot	https://github.com/STMicroelectronics/u-boot/tags
03.kernel		linux	https://github.com/STMicroelectronics/linux/tags
05.initrd	02.sysapps	busybox	https://busybox.net/downloads/
06.rootfs	01.syslibs	curl	https://curl.se/download.html
		libcap	https://git.kernel.org/pub/scm/libs/libcap/libcap.git
		libestr	https://github.com/rsyslog/libestr/tags
		libfastjson	https://github.com/rsyslog/libfastjson/tags
		libmnl	https://www.netfilter.org/projects/libmnl/downloads.html
		libpcap	https://www.tcpdump.org/release/
		libtar	https://github.com/tklauser/libtar/tags
		libtirpc	https://sourceforge.net/projects/libtirpc/
		libxml2	https://gitlab.gnome.org/GNOME/libxml2/-/tags
		lzo	http://www.oberhumer.com/opensource/lzo/download/
		popt	http://ftp.rpm.org/popt/releases/popt-1.x/
		util-linux	https://github.com/util-linux/util-linux/tags
		zlib	https://zlib.net/

arborescence		nom et version	origine des sources
	02.sysapps	btrfs-progs	https://git.kernel.org/pub/scm/linux/kernel/git/kdave/btrfs-progs.git
		busybox	https://busybox.net/downloads/
		dropbear	https://github.com/mkj/dropbear/tags
		ethtool	https://mirrors.edge.kernel.org/pub/software/network/ethtool/
		i2c-tools	https://mirrors.edge.kernel.org/pub/software/utils/i2c-tools/
		iperf	https://github.com/esnet/iperf/tags
		logrotate	https://github.com/logrotate/logrotate/tags
		mtd-utils	https://github.com/sigma-star/mtd-utils/tags
		ntp-4.2	http://www.ntp.org/downloads.html
		parted	https://ftp.gnu.org/gnu/parted/
		phytool	https://github.com/wkz/phytool/tags
		rsyslog	https://github.com/rsyslog/rsyslog/tags
		tcpdump	https://www.tcpdump.org/release/
		tzdata	https://data.iana.org/time-zones/releases/

Les versions utilisées sont indiquées dans les FCL du logiciel, cf. §1.1.