（Development of iTC）

（ATP Software Requirement Specification）

Ref-iTC/2038/

卡斯柯信号有限公司

2014年7月4日

审核页

|  |  |  |  |
| --- | --- | --- | --- |
| 拟制： |  | 日期： |  |
|  |  |  |  |
| 审核： |  | 日期： |  |
|  |  |  |  |
| 批准： |  | 日期： |  |

注：对外提交本文档时，请删除本页。

REVISIONS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Date | Updated  Version | Author | Comments |
| 1 | 2011-1-13 | V 0.0.1 | 常鸣  ChangMing | 创建全文 |
| 2 | 2011-3-10 | V0.0.2 | 常鸣  ChangMing | 根据评审意见修改全文 |
| 3 | 2011-3-10 | V1.0.0 | 常鸣  ChangMing | 正式发布文档  Formal released |
| 4 | 2011-3-26 | V1.0.1 | 常鸣  ChangMing | iTC00000156 |
| 5 | 2011-4-11 | V1.0.2 | 常鸣  ChangMing | iTC00000196 |
| 6 | 2011-4-27 | V1.0.3 | 常鸣  ChangMing | iTC00000246 |
| 7 | 2011-5-27 | V1.0.4 | 常鸣  ChangMing | iTC00000301 |
| 8 | 2011-6-2 | V1.0.5 | 常鸣  ChangMing | iTC00000325 |
| 9 | 2011-6-10 | V1.0.6 | 常鸣  ChangMing | iTC00000428, iTC00000457,iTC00000484 |
| 10 | 2011-6-21 | V1.0.7 | 常鸣  ChangMing | iTC00000567, iTC00000483 |
| 11 | 2011-7-1 | V1.0.8 | 常鸣  ChangMing | iTC00000570, iTC00000576,iTC00000577, iTC00000581  iTC00000582, iTC00000589,iTC00000590, iTC00000597 |
| 12 | 2011-7-8 | V1.0.9 | 常鸣  ChangMing | iTC00000616, iTC00000626,iTC00000628, iTC00000660 |
| 13 | 2011-7-28 | V1.0.10 | 常鸣  ChangMing | iTC00000626, iTC00000628,iTC00000710, iTC00000740  iTC00000744, |
| 14 | 2011-8-8 | V1.0.11 | 常鸣  ChangMing | iTC00000589, iTC00000710,iTC00000740, iTC00000768,  iTC00000807, iTC00000833,iTC00000835, iTC00000844, |
| 15 | 2011-8-10 | V1.0.12 | 常鸣  ChangMing | iTC00000861 |
| 16 | 2011-8-22 | V1.0.13 | 常鸣  ChangMing | iTC00000999 |
| 17 | 2011-8-29 | V1.0.14 | 常鸣  ChangMing | iTC00000999 |
| 18 | 2011-9-13 | V1.1.0 | 常鸣  ChangMing | iTC00001180  Formal release according to "ATP Software Requirement Specification Verification Report-V1.0.0" |
| 19 | 2011-9-27 | V1.1.1 | 常鸣  ChangMing | iTC00001174, iTC00001175, iTC00001231,iTC00001125,  iTC00000888, iTC00001349,iTC00001389, |
| 20 | 2011-9-29 | V1.1.2 | 常鸣  ChangMing | iTC00001457, iTC00001564 |
| 21 | 2011-9-30 | V1.2.0 | 常鸣  ChangMing | iTC00001599  Formal release according to "ATP Software Requirement Specification Verification Report-V1.1.0" |
| 22 | 2011-11-02 | V1.2.1 | 常鸣  ChangMing | iTC00001659, iTC00001682, iTC00001888, iTC00001726,  iTC00001776, iTC00001925, iTC00001967, iTC00001999,  iTC00002002, iTC00002019, iTC00002043, iTC00002112 |
| 23 | 2011-11-15 | V1.2.2 | 常鸣  ChangMing | iTC00002111, iTC00002207, iTC00002218, iTC00002220,  iTC00002270, iTC00002288, iTC00002299, iTC00002317,  iTC00002318, iTC00002319 |
| 24 | 2011-11-24 | V1.2.3 | ChangMing | iTC00001999, iTC00002002, iTC00002043, iTC00002111,  iTC00002218, iTC00002220, iTC00002317, iTC00002318,  iTC00002531, iTC00002532, iTC00002533, iTC00002575,  iTC00002594 |
| 25 | 2011-12-05 | V1.2.4 | ChangMing | iTC00002043, iTC00002317, iTC00002532, iTC00002662,  iTC00002688, iTC00002696, iTC00002759 |
| 26 | 2011-12-10 | V1.2.5 | ChangMing | iTC00002666, iTC00002838, iTC00002848, iTC00002899,  iTC00002909 |
| 27 | 2011-12-19 | V1.3.0 | ChangMing | iTC00002925  Formal release according to "ATP Software Requirement Specification Verification Report-V1.2.0" |
| 28 | 2011-12-28 | V1.3.1 | ChangMing | iTC00003027, iTC00003030, iTC00003066 |
| 29 | 2012-1-4 | V1.3.2 | ChangMing | iTC00003254, iTC00003295, iTC00003298 |
| 30 | 2012-2-1 | V1.3.3 | ChangMing | iTC00003377 |
| 31 | 2012-2-17 | V1.3.4 | ChangMing | iTC00003516, iTC00003519, iTC00003566, iTC00003608,  iTC00003609 |
| 32 | 2012-3-2 | V1.3.5 | ChangMing | iTC00003803, iTC00003694, iTC00003744, iTC00003784 |
| 33 | 2012-3-26 | V1.3.6 | ChangMing | iTC00003898, iTC00003934, iTC00003935, iTC00004262,  iTC00004293, iTC00004299 |
| 34 | 2012-3-30 | V1.3.7 | ChangMing | iTC00004377  SwRS-0587: Negate the output value of RM\_ACT1/2 |
| 36 | 2012-4-9 | V1.4.0 | ChangMing | iTC00004417    Formal release according to "ATP Software Requirement Specification Verification Report-V1.3.0" |
| 37 | 2012-4-18 | V1.4.1 | ChangMing | iTC00004421, iTC00004488, iTC00004630, iTC00004644 |
| 38 | 2012-4-25 | V1.4.2 | ChangMing | iTC00004708, iTC00004729 |
| 39 | 2012-4-28 | V1.4.3 | ChangMing | iTC00004818 |
| 40 | 2012-4-28 | V1.5.0 | ChangMing | iTC00004825  Formal release according to "ATP Software Requirement Specification Verification Report-V1.4.0" |
| 41 | 2012-5-31 | V1.5.1 | ChangMing | iTC00005244, iTC00005317 |
| 42 | 2012-6-14 | V1.5.2 | ChangMing | iTC00005493 |
| 43 | 2012-6-20 | V1.5.3 | ChangMing | iTC00005760  Table2-2, Update the version of EN50128 |
| 44 | 2012-7-4 | V1.5.4 | ChangMing | iTC00005782  1. SwRS-0159, replace the referenced source to SyAD-0290 from SyAD-0297;  2. SwRS-0114, add SwHA-0231 as the referenced source;  3. SwRS-0590, add SwHA-0232 as the referenced source. |
| 45 | 2012-7-10 | V1.6.0 | ChangMing | iTC00005797  Formal release according to "ATP Software Requirement Specification Verification Report-V1.5.0" |
| 46 | 2012-8-24 | V1.6.1 | ChangMing | iTC00006084, iTC00006086, iTC00006090, iTC00006092,  iTC00006093 |
| 47 | 2012-9-14 | V1.6.2 | ChangMing | iTC00006182, iTC00006184, iTC00006185, iTC00006272  iTC00006301 |
| 48 | 2012-10-8 | V1.6.3 | ChangMing | iTC00006272, iTC00006184, iTC00006411 |
| 49 | 2012-11-14 | V1.6.4 | ChangMing | iTC00006570 |
| 50 | 2012-11-23 | V1.7.0 | ChangMing | iTC00006639  Formal release according to "ATP Software Requirement Specification Verification Report-V1.6.0" |
| 51 | 2013-3-18 | V1.7.1 | ChangMing | iTC00006847, iTC00006942, iTC00006943, iTC00006944,  iTC00006962 |
| 52 | 2013-3-26 | V2.0.0 | ChangMing | iTC00007054: Formal release according to the review report. |
| 53 | 2013-4-18 | V2.0.1 | ChangMing | iTC00007173 |
| 54 | 2013-5-23 | V2.0.2 | ChangMing | iTC00007309 |
| 55 | 2013-5-31 | V2.0.3 | ChangMing | iTC00007341 |
| 56 | 2013-6-6 | V2.1.0 | ChangMing | iTC00007365: Formal release according to "ATP Software Requirement Specification Verification Report V2.1.0" |
| 57 | 2013-7-29 | V2.1.1 | ChangMing | iTC00007639, iTC00007697 |
| 58 | 2013-11-5 | V2.1.2 | ChangMing | iTC00007788, iTC00007842, iTC00007979, iTC00008039,  iTC00008264 |
| 59 | 2013-12-30 | V2.1.3 | ChangMing | iTC00008527, iTC00008528, iTC00008643 |
| 60 | 2013-12-31 | V2.2.0 | ChangMing | iTC00008713, formal release V220 |
| 61 | 2014-1-17 | V2.2.1 | ChangMing | iTC00008339, iTC00008521, iTC00008555, iTC00008562,  iTC00008565, iTC00008575, iTC00008593, iTC00008641,  iTC00008826 |
| 62 | 2014-2-26 | V2.2.2 | ChangMing | iTC00008833, iTC00008339, iTC00008889, iTC00008890,  iTC00008895, iTC00008923, iTC00008924, iTC00008925,  iTC00008926, iTC00008927, iTC00008928 |
| 63 | 2014-4-4 | V2.2.3 | ChangMing | iTC00008918, iTC00008926, iTC00009060, iTC00009061,  iTC00009062, iTC00009064, iTC00009066, iTC00009067,  iTC00009082 |
| 64 | 2014-4-22 | V2.2.4 | ChangMing | iTC00009064, iTC00009075, iTC00009097, iTC00009137,  iTC00009139, iTC00009148 |
| 65 | 2014-6-11 | V2.2.5 | ChangMing | iTC00009062, iTC00009425, iTC00009427, iTC00009458,  iTC00009520, iTC00009522, |
| 66 | 2014-7-4 | V2.2.6 | ChangMing | iTC00009427, iTC00009627, iTC00009724, iTC00009726  iTC00009728, iTC00009730, iTC00009745 |
|  |  |  |  |  |
|  |  |  |  |  |

CONTENTS

[1 GENERAL DESCRIPTION 16](#_Toc392229890)

[1.1 Purpose of the Document 16](#_Toc392229891)

[1.2 Field of Application 16](#_Toc392229892)

[1.3 Identification of the Last Requirement Defined 16](#_Toc392229893)

[2 SYSTEM CONTEXT 17](#_Toc392229894)

[2.1 Purpose 17](#_Toc392229895)

[2.2 Assumptions 17](#_Toc392229896)

[2.3 Application and Reference Documents 18](#_Toc392229897)

[2.4 Abbreviations and Defines 19](#_Toc392229898)

[2.5 Conventions 19](#_Toc392229899)

[3 GENERAL CONSTRAINTS 21](#_Toc392229900)

[3.1 User Characteristics 21](#_Toc392229901)

[3.2 Safety Requirements 21](#_Toc392229902)

[3.3 Production Constraints 21](#_Toc392229903)

[3.4 VCP Constraints 22](#_Toc392229904)

[3.5 Quality Requirements 28](#_Toc392229905)

[3.6 Performance 28](#_Toc392229906)

[4 INTERFACE SPECIFICATIONS 30](#_Toc392229907)

[4.1 List of Interface 30](#_Toc392229908)

[4.2 Interface with CC Data Plug 31](#_Toc392229909)

[4.3 Interface with VPB 32](#_Toc392229910)

[4.4 Interface with CC Non Vital 34](#_Toc392229911)

[4.5 Interface with Distant ATP 43](#_Toc392229912)

[4.6 Interface with VIOM 47](#_Toc392229913)

[4.7 Interface with ZC 50](#_Toc392229914)

[4.8 Interface with LC 56](#_Toc392229915)

[4.9 Interface with CI Radio 61](#_Toc392229916)

[4.10 Interface with PSD 65](#_Toc392229917)

[4.11 Interface with DLU 70](#_Toc392229918)

[4.12 Interface with Project Data 71](#_Toc392229919)

[4.13 Interface between two CPUs 72](#_Toc392229920)

[4.14 Interface with Memorized Location 75](#_Toc392229921)

[4.15 Interface with VLE Hardware 76](#_Toc392229922)

[5 FUNCTIONAL REQUIREMENTS 78](#_Toc392229923)

[5.1 Description of Functions 78](#_Toc392229924)

[5.2 Description of Data 79](#_Toc392229925)

[5.3 F1- Manage System Information 82](#_Toc392229926)

[5.4 F11-Acquire Configuration Data 82](#_Toc392229927)

[5.5 F12-Manage Train Status 89](#_Toc392229928)

[5.6 F13-Manage Loop Hour with Distant ATP 101](#_Toc392229929)

[5.7 F14-Manage Variants in Block Mode 109](#_Toc392229930)

[5.8 F15-Manage Variants in CBTC Mode 121](#_Toc392229931)

[5.9 F2-Measure Train Kinematics 131](#_Toc392229932)

[5.10 F21-Manage Interface with VPB 132](#_Toc392229933)

[5.11 F22-Monitor the Odometer 144](#_Toc392229934)

[5.12 F23-Manage the Odometer State 152](#_Toc392229935)

[5.13 F24-Compensate Sliding Slipping Effect 163](#_Toc392229936)

[5.14 F25-Calculate Radar Speed 187](#_Toc392229937)

[5.15 F26-Detect Odometer Axle Lock 190](#_Toc392229938)

[5.16 F27-Compute Train Kinematics 201](#_Toc392229939)

[5.17 F28-Calibrate Wheel Movement 209](#_Toc392229940)

[5.18 F3-Locate the Train on Track Map 217](#_Toc392229941)

[5.19 F31-Initialize Train Location 218](#_Toc392229942)

[5.20 F32-Update Train Location 228](#_Toc392229943)

[5.21 F33-Confirm Train Localization 237](#_Toc392229944)

[5.22 F4-Monitor Train Energy 249](#_Toc392229945)

[5.23 F41-Determine the EOA 249](#_Toc392229946)

[5.24 F42-Manage Coerced Permissive or Restrictive 259](#_Toc392229947)

[5.25 F43-Manage Temporary Speed Restriction 266](#_Toc392229948)

[5.26 F44-Compute Train Energy 270](#_Toc392229949)

[5.27 F45-Process Singularities 274](#_Toc392229950)

[5.28 F46-Determine Over Energy 296](#_Toc392229951)

[5.29 F5-Monitor Train Position and Speed 299](#_Toc392229952)

[5.30 F51-Moral Time Control 300](#_Toc392229953)

[5.31 F52-Prevent Train Moving in Undetectable Danger 307](#_Toc392229954)

[5.32 F53-Monitor Train Speed 317](#_Toc392229955)

[5.33 F54-Monitor Rollback Train Speed 319](#_Toc392229956)

[5.34 F55-Monitor Reverse Train Speed 325](#_Toc392229957)

[5.35 F6-Protect Passengers Entrance and Exit from the Train 330](#_Toc392229958)

[5.36 F61-Elaborate Door Opening Authorization 331](#_Toc392229959)

[5.37 F62-Control PSD Opening and Closing Order 339](#_Toc392229960)

[5.38 F63-Monitor Status of Doors 353](#_Toc392229961)

[5.39 F64-Protect Passengers during Emergency Evacuation 363](#_Toc392229962)

[5.40 F7-Generate Output Orders 373](#_Toc392229963)

[5.41 F71-Outputs to Rolling-stock 374](#_Toc392229964)

[5.42 F72-Outputs to ZC 385](#_Toc392229965)

[5.43 F73-Outputs to CI Radio 403](#_Toc392229966)

[5.44 F8-Platform Relative & Assist Functions 405](#_Toc392229967)

[5.45 F81-Initialize ATP software 405](#_Toc392229968)

[5.46 F82-Manage Vital Time 409](#_Toc392229969)

[5.47 F83-Manage Synchronization between two CPUs 417](#_Toc392229970)

[6 Appendices 424](#_Toc392229971)

[6.1 Project Configuration of ATPsetting 424](#_Toc392229972)

[6.2 Carborne Controller Constants 427](#_Toc392229973)

[6.3 Logical Types Definition 430](#_Toc392229974)

[6.4 Offline generated codes 435](#_Toc392229975)

[6.5 Message 436](#_Toc392229976)

[6.6 Track Map 439](#_Toc392229977)

[6.7 Assessment on Compliance with EN50128 446](#_Toc392229978)

FIGURES

[Figure 4‑1 CC-CBI communication with SACEM protocol 62](#_Toc392229979)

[Figure 4‑2 CC-CBI communication with FSFB2 protocol 67](#_Toc392229980)

[Figure 5‑1 Track map layout for project 80](#_Toc392229981)

[Figure 5‑2 Abscissa increasing rule in block 81](#_Toc392229982)

[Figure 5‑3 SART modeling of function F1 82](#_Toc392229983)

[Figure 5‑4 Configurable Rolling-stock Inputs 96](#_Toc392229984)

[Figure 5‑5 Train Coupled Status 98](#_Toc392229985)

[Figure 5‑6 SART modeling of function F2 131](#_Toc392229986)

[Figure 5‑7 Odometer state 154](#_Toc392229987)

[Figure 5‑8 Processing of over estimation state 169](#_Toc392229988)

[Figure 5‑9 Processing of under estimation state 179](#_Toc392229989)

[Figure 5‑10 Processing of calibration 211](#_Toc392229990)

[Figure 5‑11 SART modeling of function F3 218](#_Toc392229991)

[Figure 5‑12 Determine orientation by two neighbor beacons 220](#_Toc392229992)

[Figure 5‑13 The balloon loop 221](#_Toc392229993)

[Figure 5‑14 Train location with orientation 223](#_Toc392229994)

[Figure 5‑15 Train localization state 243](#_Toc392229995)

[Figure 5‑16 SART modeling of function F4 249](#_Toc392229996)

[Figure 5‑17 Train located in BM initialZone 252](#_Toc392229997)

[Figure 5‑18 PSR as vital speed limit zone 277](#_Toc392229998)

[Figure 5‑19 SART modeling of function F5 300](#_Toc392229999)

[Figure 5‑20 SART modeling of function F6 331](#_Toc392230000)

[Figure 5‑21 SART modeling of function F7 374](#_Toc392230001)

[Figure 5‑22 SART modeling of function F8 405](#_Toc392230002)

[Figure 5‑23 Synchronize cycle sequence between two CPUs 415](#_Toc392230003)

[Figure 5‑24 Manage synchronization between two CPUs 418](#_Toc392230004)

[Figure 6‑1 Neighboured beacon principle in ATP 442](#_Toc392230005)

TABLES

Table 2‑1 Applicable documents 18

Table 2‑2 Reference documents 18

Table 2‑3 Description of ARDL common functions 20

Table 4‑1 List of external interface 30

Table 4‑2 List of internal interface 30

Table 4‑3 List of data from CC data plug 31

Table 4‑4 List of data from CC data plug 32

Table 4‑5 Data send to VPB 33

Table 4‑6 List of data from VPB 34

Table 4‑7 Content of CCNV initialization report 36

Table 4‑8 ATP inputs from CCNV 37

Table 4‑9 Structure of ATP non-vital report 38

Table 4‑10 ATP outputs to redundant ATP 44

Table 4‑11 ATP inputs from VIOM 48

Table 4‑12 Structure of ATP outputs to VIOM 49

Table 4‑13 ATP outputs to *ZC* 51

Table 4‑14 EOA report from *ZC* 53

Table 4‑15 Variant report from *ZC* 55

Table 4‑16 Date synchronization report from LC 57

Table 4‑17 TSR download content from LC 58

Table 4‑18 Version authorization from LC 59

Table 4‑19 ATP outputs to LC 60

Table 4‑20 CBI variant request 63

Table 4‑21 CBI variant report 64

Table 4‑22 CC variant request 64

Table 4‑23 CC variant report 65

Table 4‑24 PSD control 67

Table 4‑25 IO status from CI 69

Table 4‑26 ATP output to DLU 71

Table 4‑27 ATP offline tool inputs from project data 72

Table 4‑28 Synchronization data write to the other CPU 73

Table 4‑29 Synchronization data read from the other CPU 74

Table 4‑30 Memorized location content 75

Table 4‑31 ATP inputs from VLE-2 76

Table 5‑1 Functions table 78

Table 5‑2 Constants for iTC production 79

Table 5‑3 Index of variant 81

Table 5‑4 Identical VIOM Inputs for offline application 94

Table 5‑5 Configurable Inputs from Rolling Stock 97

Table 5‑6 Odometer information in interrupt 133

Table 5‑7 Configurable not coerced restrictive identification 262

Table 5‑8 Configurable coerced permissive identification 264

Table 5‑11 Backward distance account rules 326

Table 5‑13 Configuration for monitoring unexpected train door open 357

Table 6‑2 Carborne Controller Constants 427

Table 6‑3 Get combined inputs from offline generated codes 435

# GENERAL DESCRIPTION

## Purpose of the Document

本文档用于定义卡斯柯信号有限公司智能列车控制系统车载控制器子系统中分配给ATP软件的需求描述，包括ATP软件需要遵从的产品约束，安全要求，可靠性，可维护性，接口规范以及所有功能需求等方面的规格定义。

This document defines the ATP software requirements assigned from the onboard train control subsystem of the CASCO's intelligent train control production, including the constraints, safety requirements reliability, maintainability, interface specifications and all functional requirements with which ATP software need to comply.

本文档所述ATP软件的范围包括运行在车载控制器机笼内VLE-2板卡上的ATP在线软件，以及支持ATP软件运行所需的通信接口。

The ATP software described in this document is not only including the software running on the VLE-2 board deployed in the onboard controller rack, but including the communication interface with other systems.

本文档是ATP软件开发人员的设计依据，也是安全和测试人员编写相关文档的参考文档，供项目经理，系统需求分析人员审阅。

This document is the specification for the ATP software designers, and is the reference for safety analysts, valuators, project managers and requirement verifiers etc.

## Field of Application

本文档是iTC研发的一部分，属于iCMTC信号系统解决方案车载控制器子系统。This document is part of iTC R&D, and it concerns the Carborne Controller, which is an onboard ATC part of the iCMTC signaling solution.

## Identification of the Last Requirement Defined

The last requirement defined is the ATP-SwRS-0804, and the numeration must restart from ATP-SwRS-0805.

# SYSTEM CONTEXT

## Purpose

ATP软件是车载控制器的核心组成部分，起到保障列车运行安全的重要作用。ATP软件所实现的功能包括：

* 利用板卡硬件资源，进行时间周期控制和二取二数据比对；
* 获取来自外部系统或CC内部其他子系统的数据；
* 通过来自车辆及其内外部系统的数据，判断ATP自身的工作状态、线路前方的变量状态以及授权终点等；
* 根据里程计的测速信息以及轨旁信标信息，计算列车的运动学信息以及在线路地图中的位置；
* 监控列车的速度、位置和能量不超过环境限制，确保列车能在授权终点前停车；
* 根据站台信息，授权指定侧车门开启；
* 生成对外部系统和车辆的安全输出信息。

The ATP software is the core component of the Carborne Controller, playing the important role for guaranteeing the safety of the train during operation. ATP achieved including:

* using hardware board resources to realize the period time control, and the 2oo2 data comparison;
* access external systems or data from other subsystems within the CC;
* through from the rolling stock and internal or external systems data to determine the state of ATP, the EOA and dynamic track profiles;
* according to the odometer and trackside beacon message, calculating the train's kinematics and localizing the train on the map;
* monitoring train speed, location and energy to avoid the train to exceed the environmental restricted, and ensuring the train to stop before the end of authorized;
* according to the platform information, enabling to operate the train door and PSD;
* generating vital reports and commands to external systems and the rolling stock.

## Assumptions

ATP软件开发人员应当了解城市轨道交通信号系统运营的基本原理，熟悉车载控制器的工作环境和所实现的功能，了解机笼内各板卡的硬件结构和所实现功能，具有实时嵌入式系统的开发经验，遵循铁路信号系统软件开发的相关标准。ATP software developers should be aware of the basic principles of urban rail transit signaling system, familiar with carborne controller's requirements and work environment, understand the hardware structure and functions of each board, experience with real-time embedded systems development, and follow the relative standards of the railway signaling production.

ATP软件的安全部分应用“安全编码处理器”技术，该技术保证在使用专用的工具处理过ATP的源代码后，能将其转换为冗余编码形式，使得在运算过程中发生的硬件随机故障能以符合SIL4要求的概率导致冗余编码错误，从而被CC的检测机制检查出来。

The vital part of ATP software applies "Vital Coded Processor " technique, which can convert the raw ATP source codes to the vital coded codes, and guarantee that the occur of random failures by the hardware must lead to the vital code error under the probability of SIL4, and the vital code errors can be detected by the checking mechanism.

车载ATP软件运行在卡斯柯新型2乘2取2安全计算平台上，硬件平台应具有足够的资源，满足ATP软件计算的实时性要求。同时，该硬件平台能提供2乘2冗余通信条件，以及2取2双通道计算。

The onboard ATP software running on CASCO's new 2×2oo2 vital computation platform. The hardware platform should have sufficient resources to meet the performance requirements of ATP software. Meanwhile, the platform should provide 2 channels for redundant communication and independent computation.

## Application and Reference Documents

Table 2‑1 Applicable documents

|  |  |  |
| --- | --- | --- |
| Document title | Version | Reference |
| 1. iTC System glossary | V1.1.0 | iTC-0001 |
|  |  |  |

Table 2‑2 Reference documents

|  |  |  |
| --- | --- | --- |
| Document title | Version | Reference |
| 1. CC Subsystem Requirement Specification | V2.0.0 | iTC-2026 |
| 1. CC Subsystem Architecture Description | V2.0.0 | iTC-3021 |
| 1. ATP-Data Plugger Interface description | V2.0.0 | iTC-3023 |
| 1. VLE-2-VPB-2 Interface description | V2.1.0 | iTC-3024 |
| 1. VLE-2-DVCOM-2 Interface description | V2.1.0 | iTC-3022 |
| 1. CC-Redundant CC Interface description | V2.1.0 | iTC-3025 |
| 1. CC-ZC Interface description | V2.0.0 | iTC-3002 |
| 1. CC-LC Interface description | V2.0.0 | iTC-3003 |
| 1. VLE-2 board user manual | V1.1.0 | 2oo2-601 |
| 1. Train Energy Monitoring Principle | V1.1.0 | iTC-3040 |
| 1. ATP-CC offline tool interface description | V2.1.0 | iTC-3019 |
| 1. VLE-2-DLU Interface description | V2.0.0 | iTC-3051 |
| 1. VCP1 Software User Manual | V1.4.0 | iCoder-501 |
| 1. VCP2 Software User Manual | V1.4.0 | iCoder-501 |
| 1. iCODER Software Supported C-Language Subset Coding Specification | V1.4.0 | iCoder-503 |
| 1. Railway applications - communication, signalling and processing systems - software for railway control and protection systems | 2001 | EN50128 |
| 1. Railway applications - communication, signalling and processing systems | 2010 | EN50159 |

## Abbreviations and Defines

Reference to [APP1] for abbreviations and defines.

缩写词和定义参见[APP1]。

## Conventions

ATP软件的开发流程遵循EN50128标准中所定义的SIL4级软件要求。ATP软件的需求继承自CC子系统架构描述中分配给VLE板软件部分，并定义了唯一的需求标签与之追溯。在软件需求定义文档中，ATP软件依照功能划分为不同模块，对于每个模块均定义了输入输出变量。通常，对于每条功能性需求，均明确定义了单独的变量，以此可以通过维护工具进行观察，判断该需求是否正确执行。为增加软件需求的精确性，本文档定义并使用 “精确需求描述语言”(ARDL, Accurate Requirements Description Language)对需求过程进行描述，便于软件设计、测试和验证人员阅读。

The organization of ATP software development shall follow the process of SIL4 system defined in EN50128. The requirements of ATP inherit from the CC subsystem architecture description, and defining a unique label for each requirement is used to guarantee the traceability with the higher level demands. In this specification, ATP requirements are partitioned as different functional modules; and for each module, the inputs and outputs shall be defined clearly. Each functional requirement of ATP shall be testable, which means the document shall give a precise definition for each requirement. Therefore, for increasing the accuracy, ARDL (Accurate Requirement Description Language) are defined and used to describe the procedure of a requirement. In this way, tester easy to set test conditions according to the ARDL and observe whether the requirement is realized correctly.

需求描述中所用ARDL的约定如下：

* 使用缩进表示层次关系；
* 每个变量后的(k)表示当前周期的值，而(k-1)等表示上个周期或之前的值；
* 使用"and", "or", 和"not" 表示逻辑的“与”，“或"，“非”操作。
* 数组使用中括号表示；
* 常数使用大写标识。

The conventions of ARDL used for describing software requirements are shown as following:

* Use indentation to express hierarchy structure;
* Use (k) after each variable to represent the value in the current cycle, and (k-1) for the last cycle;
* Use “and”, “or”, “not” for logic operation;
* Use brackets to represent arrays;
* Use uppercase to represent system constants.

在ARDL中使用的一些通用函数的含义如Table 2‑3所示：

The description of common functions used in ARDL are shown as Table 2‑3.

Table 2‑3 Description of ARDL common functions

|  |  |  |
| --- | --- | --- |
| Function | Description | Example |
| abs(A) | Return the absolute value of A | abs(-1) = 2 |
| max(A,B) | Return the maximum value of two numeric A or B | max(-3, 0) = 0 |
| min(A, B) | Return the minimum value of two numeric A or B | min(0, 5) = 0 |
| pow(A) | Return the square of A | pow(5) = 25 |
| range(A, B) | Return a list of natural numbers from A to B-1 | range(0, 3) = {0, 1, 2} |
| round.ceil(A) | Round up to the nearest integer. | round.ceil(3.1) = 4 |
| round.floor(A) | Round down to the nearest integer. | round.floor(5.8) = 5 |
| sign(A) | Return the positive or negative sign of the variable | sign(5)=+1, sign(-6)=-1, sign(0)=0 |

# GENERAL CONSTRAINTS

## User Characteristics

N/A

## Safety Requirements

[iTC\_CC\_ATP-SwRS-0001]

车载ATP软件应按照[REF16]，[REF17]标准中所定义的SIL4系统进行开发，见6.7，并达到SIL4要求。

Refer to 6.7, the onboard ATP software shall be SIL4 according to the CENELEC standard [REF16] and [REF17].

#Category=Non-Functional

#Contribution=Safety

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0030], [iTC\_CC-SyAD-0828]

[End]

## Production Constraints

[iTC\_CC\_ATP-SwRS-0003]

ATP软件使用的配置参数来自iTC系统数据准备流程，对于每个工程项目，ATP软件可读取相应的线路地图和安全配置参数，从而适应不同项目中对列车特性和系统设计的要求。

ATP software shall use parameters come from iTC system data preparation for different project's deployment. For each project, ATP software reads the track map and vital setting to adapt the features of train and system design.

#Category=Non-Functional

#Contribution=N/A

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0995]

[End]

[iTC\_CC\_ATP-SwRS-0004]

由于系统设计限制，ATP通过CCNV最多同时与2套PSD设备进行通信。

For sub-system design constraints, it is not possible to establish more than two channel of communication with PSD platform manager. Therefore, ATP shall activate a maximum of two communication channels with PSD platform managers ordered by CC Non Vital.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0262]

[End]

[iTC\_CC\_ATP-SwRS-0005]

ATP使用FSFB2安全通信协议与PSD进行通信。

For interface design constraints, ATP shall use FSFB2 safety protocol with PSD platform managers with respect with safety principles.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0259], [iTC\_CC\_ATP\_SwHA-0168]

[End]

[iTC\_CC\_ATP-SwRS-0550]

在运行过程中，ATP软件不响应任何来自VLE-2板前网口的连接请求和命令，防止非法数据攻击。

During the operation, the onboard ATP shall not response any request or command from the VLE-2 debugging Ethernet port, in order to avoid the invalid accesses and the attacks.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0063], [iTC\_CC-SyAD-0113]

[End]

## VCP Constraints

ATP使用VCP工具iCODER-100编码，且双CPU采用不同的VCP工具iCODER100-VCP1和iCODER100-VCP2，因此应需遵循[REF13]和[REF14]的设计限制，具体如下所示：

[iTC\_CC\_ATP-SwRS-0498]

车载ATP程序的安全部分使用符合安全处理器（VCP）规则[REF15]的C语言编写。

The codes of the onboard ATP shall follow the [REF15] rules of the VCP.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0010], [iTC\_CC-SyAD-0044], [iTC\_CC-SyAD-0115], [iCODER100-VCP1-SwUR-0001], [iCODER100-VCP2-SwUR-0001]

[End]

**NOTES**:

经过VCP编码工具处理后的ATP安全相关变量，其高32位为数据位，低32位为校验位，校验位包含了时间标签技术。

For each vital variable encoded by the VCP tool, its higher 32 bits are raw data, and its lower 32 bits are check words of the higher data. The check words involve time-label technique.

[iTC\_CC\_ATP-SwRS-0499]

ATP中安全相关的运算采用VCP编码，在安全运算过程中引入非安全相关运算中产生的变量时，必须先对该变量进行VCP编码。包括项目配置参数，线路地图，CC data plug等来自外部的数据，均需离线进行VCP编码。

For VCP constraints, the onboard ATP importing data from non-vital devices shall encode firstly. The data of project configuration, track map and CC data plug, all coded by the VCP offline tool.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0011], [iTC\_CC-SyAD-0008]

[End]

[iTC\_CC\_ATP-SwRS-0500]

每套车载ATP软件由分别运行在VLE-2板两个CPU中各自独立的ATP程序组成，它们分别采用的iCODER100-VCP1和iCODER100-VCP2进行编码，形成2取2结构。

Each set of onboard ATP shall have two parts of programs, which compose two out of two structures. One program deploys on one of two CPUs of the VLE-2 board, and the other deploy on another CPU. They perform similar functions but encode by iCODER100-VCP1 and iCODER100-VCP2 respectively.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0002], [iTC\_CC-SyAD-0006],[iTC\_CC-SyAD-0116], [iTC\_CC-SyAD-0828]

[End]

[iTC\_CC\_ATP-SwRS-0501]

运行车载ATP程序的操作系统为VxWorks 5.5。

The onboard ATP shall work on the VxWorks 5.5 operating system.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0062], [iCODER100-VCP1-SwUR-0009], [iCODER100-VCP2-SwUR-0009]

[End]

[iTC\_CC\_ATP-SwRS-0760]

ATP使用的iCODER-VCP1和iCODER-VCP2应装在两个不同的计算机上。独立运行VCP1和VCP2生成的两份冗余代码只有在经过比较且一致后才能使用。用于双链冗余代码比较的工具是两个不同的商用比较工具。

iCODER-VCP1 and iCODER-VCP2 shall be installed in two different computer. The two redundant codes generated by independent VCP1 and VCP2 must be fit through comparison before being used. Two different commercial compare tools have been used to compare the redundant code generated by two channel.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iCODER100-VCP1-SwUR-0002], [iCODER100-VCP2-SwUR-0002]

[End]

[iTC\_CC\_ATP-SwRS-0761]

ATP在源码中需要进行冗余编码的全局变量的个数（结构体全局变量以结构体的成员个数计算）不能超过3000个，函数个数不能超过300个，函数内的局部变量个数不能超过20个（结构体局部变量以结构体的成员个数计算），全局和局部布尔变量的个数不能超过100个。

The number of global variables that need to be redundant coded in the ATP source code cannot be more than 3000 and the function number is less than 300 and the local variables less than 20 (the structure local variables can be calculated by the member numbers of the struct) and the Boolean variables less than 100.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source= [iCODER100-VCP1-SwUR-0003], [iCODER100-VCP2-SwUR-0003]

[End]

[iTC\_CC\_ATP-SwRS-0762]

冗余代码生成功能能够对符合以下调用顺序要求的函数进行冗余编码：模块函数（不包括注释关键字/\*KWNOPARMF\*/注释的模块函数，这类模块函数当做复杂语句来处理）只能出现在最顶层调用，并且模块函数的编码顺序必须同其在源码中的执行顺序一致。如果在运行阶段的模块函数调用顺序为fun1，接着fun2，那么在编码时也应该是先对fun1进行编码，然后对fun2进行编码。

语句与模块函数调用的编码顺序也是如此，如果在运行阶段的调用顺序为statement1，接着fun1，那么在编码时也应该是先对statement1进行编码，然后对fun1进行编码。

The redundant code generation function can generate redundant code for functions which satisfies the following constrains of calling order: the module function (Not include the module function which noted by comment key word /\*KWNOPARMF\*/, this kind of module function is processed as compound statement.) can only be called at the top level of the program, and the coded order of the module function must be the same with the execution order in source code. If the order of module function call at running phase is fun1 first and then fun2, then the redundant coding order shall also first code to fun1, then encode fun2.

The encode order between statement and module function also follow the above rules. If the calling order is statement1, and then fun1 at running time, we will encode statement1 first, then fun1 while redundant coding.#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iCODER100-VCP1-SwUR-0004], [iCODER100-VCP2-SwUR-0004]

[End]

[iTC\_CC\_ATP-SwRS-0763]

ATP代码中使用iCoder100支持的注释关键字：

The following four annotations of key words are supported:

|  |  |
| --- | --- |
| Annotation of key words | Function |
| /\*KWNOADJ\*/ | 表示此变量为NOADJ类型的变量，在全局初始化函数red\_global\_para\_init中不需要对此类变量进行调整。It means the variable is NOADJ type. This kind of variable need not be adjusted when the global initializes the function red\_global\_para\_init. |
| /\*KWNODT\*/ | 表示此变量为NODT类型的变量，即不带时间戳的变量。原则上，这类变量不应该作为左值出现在赋值运算中。在全局初始化函数red\_global\_init中也不需要对此类变量进行调整。It means the variable is NODT type, which is the variable without timestamp. In principle, this kind of variable cannot occur in the assignment operation as left value. When the global initializes the function red\_global\_init, it doesn’t need to be adjusted. |
| /\*KWCODEDBOOL\*/ | 表示此变量为布尔变量。It means the variable is the Bloom type. |
| /\*KWNOPARSE\*/ | 两条这样的注释之间包含的内容应该原样输出，不进行冗余编码。The content between these two annotations shall be output as original shape, which doesn’t make redundant coding. |
| /\*KWNOPARMF\*/ | 使用注释关键字/\*KWNOPARMF\*/注释的函数定义，表示在该函数被调用的地方，会被替换成该函数体的内容。The function definition which noted by comment key word /\*KWNOPARMF\*/, means that the function call will be replaced by the function body. |
| /\*KWADJMF\*/ | 用注释关键字/\*KWADJMF\*/注释的函数调用，表示该函数原型被单独编码。The function calling which noted by comment key word /\*KWADJMF\*/, means that the function will be coded independently. |

说明Note：

1、使用注释关键字/\*KWNOPARMF\*/注释的函数定义，表示在该函数被调用的地方，会被替换成该函数体的内容；使用限制如下：

* 该函数必须在第一次被调用之前被定义，否则冗余代码签名检查报错；
* 该函数不能有参数，否则报错；
* 该函数中不能使用return语句，否则报错；
* 该函数中不能定义局部变量，否则报错。

1.The function definition which noted by comment key word /\*KWNOPARMF\*/, means that the function call will be replaced by the function body. And use restrictions for comment key word /\*KWNOPARMF\*/ are as following:

The function must be defined before it is firstly be called, otherwise, give an error message when redundant code signature checking;

The function cannot have parameters. Otherwise, give an error message.

Return statement cannot be used in this function. Otherwise, give an error message.

Local variables cannot be used in this function. Otherwise, give an error message.

2、用注释关键字/\*KWADJMF\*/注释的函数调用，表示该函数原型被单独编码，它的使用有以下限制：

* 被注释的模块函数要满足模块函数本身的限制条件，否则报错；
* 被注释关键字/\*KWADJMF\*/注释的模块函数必须单独编码，否则冗余代码签名检查报错；
* /\*KWADJMF\*/只能用来注释模块函数调用，且编码该模块函数时，其中的全局变量在函数入口处要使用预先分配好的签名，函数中每一步运算之后和调整都要使用预先分配的签名表中的签名；如果该函数未配置，则报错；
* 所有被注释关键字/\*KWADJMF\*/注释的模块函数中用到的签名总个数不能超过5000，否则报错；
* 该模块函数中不能使用数组，否则报错；
* 该模块函数在待编码文件中只能被调用一次，否则报错。

2.The use restrictions for comment keyword /\*KWADJMF\*/ are as following.

The module function which be noted must satisfy the restrictions of module function. Otherwise, give an error message.

The module function which be noted by keyword /\*KWADJMF\*/ must be coded separately. Otherwise, give an error message when redundant code signature checking.

/\*KWADJMF\*/ can only be used to note module function call. And when coding this module function, the global variables of the module function must use the pre-assigned signatures at the entrance of the function. And after every operation and adjustment, the signature of the variable must adjust to the signature which in the pre-assigned signature table. If the function is not be configured, give an error message.

The total number of signatures which used in the module function noted by keyword /\*KWADJMF\*/ cannot exceed 5000. Otherwise, give an error message.

This module function cannot use array. Otherwise, give an error message.

This module function can only use once in the coding files. Otherwise, give an error message.

3、这些关键字只有在遇到同名的第一个时开始有效，而遇到第二个时变为无效，而与不同关键字之间的嵌套关系无关。但注释型关键字/\*KWCODEDBOOL\*/不能与/\*KWNOADJ\*/、/\*KWNOPARSE\*/或/\*KWNODT\*/嵌套使用；注释型关键字/\*KWNOPARSE\*/不能与/\*KWNOADJ\*/、/\* KWCODEDBOOL \*/或/\*KWNODT\*/嵌套使用，否则软件将报错。即两个注释关键字/\*KWCODEDBOOL\*/或/\*KWNOPARSE\*/之间不能出现注释关键字/\*KWNOADJ\*/或/\*KWNODT\*/，两个注释关键字/\*KWNOADJ\*/或/\*KWNODT\*/之间也不能出现/\*KWCODEDBOOL\*/或/\*KWNOPARSE\*/。

3.These fields annotated by keyword are only valid when we first encounter the keyword name, and become invalid when encounter the second one, and unrelated with different nesting relationship between the independent keyword. But the comment type keyword /\*KWCODEDBOOL\*/ can not nest used with /\*KWNOADJ\*/, /\*KWNOPARSE\*/ or /\*KWNODT\*/. And the comment keyword /\*KWNOPARSE\*/ cannot nest with /\*KWNOADJ\*/, /\*KWCODEDBOOL\*/ or /\*KWNODT\*/. Otherwise the iCODER-100 will make an error report.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iCODER100-VCP1-SwUR-0006], [iCODER100-VCP2-SwUR-0006]

[End]

[iTC\_CC\_ATP-SwRS-0764]

ATP将带VCP编码的运算结果发送给VIOM进行校验。

For the system which using iCODER-100 to achieve SIL requirement, the signature of the output shall be critically checked.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iCODER100-VCP1-SwUR-0007], [iCODER100-VCP2-SwUR-0007]

[End]

[iTC\_CC\_ATP-SwRS-0765]

ATP代码中头文件的嵌套不能超过十层。

Header file’s nesting cannot exceed 10 layers.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source= [iCODER100-VCP2-SwUR-0005]

[End]

[iTC\_CC\_ATP-SwRS-0766]

使用VCP工具处理ATP代码的PC机应满足的硬件最小配置：

The minimal configuration of the hardware that the software needs:

* 计算机CPU： P4 3.0GHz以上；Computer CPU: at least P4 3.0GHz
* 内存：1G以上；memory: at least 1G
* 硬盘：80G；hard disk: 80G
* 操作系统：Windows XP及以上版本的微软视窗操作系统；Operation system: Windows Xp or Microsoft windows operation system exceeds XP.

#Category=Design constraint

#Contribution=N/A

#Allocation=ATP Software

#Source=[iCODER100-VCP1-SwUR-0008], [iCODER100-VCP2-SwUR-0008]

[End]

## Quality Requirements

N/A

## Performance

[iTC\_CC\_ATP-SwRS-0008]

ATP软件必须在2个**ATP\_CYCLE\_TIME**主周期内，根据来自CCNV的输入信息计算生成外部输出命令，并组帧发送给CCNV。

Vital inputs of ATP embedded software shall be acquired every **ATP\_CYCLE\_TIME**.

Once vital inputs acquired, ATP embedded software shall refresh its outputs in less than two **ATP\_CYCLE\_TIME**, and send them to CCNV.

#Category=Non-Functional

#Contribution=Performance

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0014], [iTC\_CC-SyAD-1004], [iTC\_CC-SyAD-1006]

[End]

[iTC\_CC\_ATP-SwRS-0575]

ATP软件包烧录到VLE-2板上的时间应当不超过**C\_TTIS**分钟。

The set-up of the ATP software shall not exceed **C\_TTIS** minutes.

#Category=Non-Functional

#Contribution=Performance

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0023]

[End]

[iTC\_CC\_ATP-SwRS-0009]

ATP软件从上电到输出允许状态的信息，应当在**INIT\_AVAIL\_MAX\_TIME**时间内完成。

From power-up and if conditions on inputs allow to reach a higher level of train availability (that is to compute at least one safety-related output at permissive state) ATP embedded software shall be able to increase train operation availability in less than **INIT\_AVAIL\_MAX\_TIME**.

#Category=Non-Functional

#Contribution=Performance

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0015]

[End]

# INTERFACE SPECIFICATIONS

## List of Interface

### External interface

根据[REF2]，运行在VLE板上的ATP软件与外部系统有如下外部接口，如所示。

According to [REF2], the onboard ATP shall have external interfaces with other system, as shown in the .

Table 4‑1 List of external interface

|  |  |
| --- | --- |
| ID | Comments |
| EX1 CC Data plug | Interface with Data plug |
| EX2 VPB | 通过VPB板获取里程计和信标相关信息 |
| EX3 CC Non Vital | 通过DVCOM板与CCNV交互非安全相关信息 |
| EX4 Distant ATP | 通过DVCOM板与冗余ATP交互信息 |
| EX5 VIOM | 通过DVCOM板与VIOM交互车辆信息 |
| EX6 ZC | 通过DVCOM板与ZC交互信息 |
| EX7 LC | 通过DVCOM板与LC交互信息 |
| EX8 CI Radio | 通过DVCOM板与联锁交互增强型后备信息 |
| EX9 PSD | 通过DVCOM板与联锁交互PSD信息 |
| EX10 DLU | 与DLU交互信息 |
| EX11 Project data | Interface with project data |

### Internal interface

由于VLE-2板内部结构以及ATP程序要求，需要有如下内部接口，如Table 4‑2所示。

According to the VLE-2 structure and 2oo2 constraints, the onboard ATP shall have two internal interfaces, as shown in the Table 4‑2.

Table 4‑2 List of internal interface

|  |  |
| --- | --- |
| ID | Comments |
| IN1 Interface between 2 CPUs | Interface between two ATP module |
| IN2 Memorized Location | 获取与设置记忆定位信息 |
| IN3 VLE hardware resource | Interface with VLE library functions |

## Interface with CC Data Plug

### Role of interface

ATP通过本接口获取来自CC的设备号SSID，列车类型及所在驾驶室标识，用于判断ATP当前所在的列车类型，车头编号以及用于在网络通信中的标识信息。

ATP through this interface to obtain the device number of the CC, train type and cab id, used to determine train characteristics as well as the network identification information.

[iTC\_CC\_ATP-SwRS-0551]

ATP与CC data plug 的接口应当遵循文档[REF3]描述。其中安全数据带有VCP编码，非安全数据无需编码，如Table 4‑4所示。

The interface between ATP and CC data plug shall be compliant with [REF3] document.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0784], [iTC\_CC\_ATP\_SwHA-0185]

[End]

### Physical level

见文档[REF3]描述。

Refer to [REF3].

### Protocol level

见文档[REF3]描述。

Refer to [REF3].

### Application level

#### ATP->CC data plug

当列车连挂状态改变后，ATP软件会将新的列车类型写入Dataplug。

Table 4‑3 List of data from CC data plug

|  |  |  |
| --- | --- | --- |
| Identification | VCP coded | Comments |
| [TrainType](#TrainType) | √ | New train type if the coupled status changed |

#### CC data plug->ATP

ATP读取来自CC data plug的信息[CCdataPlugInfo](#CCdataPlugInfo)，其结构如Table 4‑4所示，其中ATP使用的安全相关信息有VCP编码，而发送给CCNV或DLU的IP地址信息无需编码。

ATP read [CCdataPlugInfo](#CCdataPlugInfo) from the CC data plug whose structure as shown in Table 4‑4. The safety-related information used by ATP are coded by VCP and the other information such IP address sent to DLU or CCNV are not need to encode.

Table 4‑4 List of data from CC data plug

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | VCP coded | Comments |
| ST\_DATA\_PLUG | |  |  |
|  | .CcSSID | √ | CC sub-system identification in communication |
|  | .CcTrainType | √ | Train type |
|  | .CcCoreId | √ | Whether ATP deployed on **END\_1** or **END\_2** of the train |
|  | .CcNetInfo | × | CC's network configuration |
|  | .VLECpuId | √ | Whether the ATP deployed on CPU1 or CPU2 of VLE |
|  | .CCNVNetInfo | × | CCNV 's network configuration |
|  | .DLUNetInfo | × | DLU's network configuration |

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with VPB

### Role of interface

ATP通过本接口获取来自VPB板实时更新的编码里程计和信标天线数据，并在停车时，将里程计传感器测试序列发给VPB板。

Through this interface, ATP obtains the real-time odometer and beacon information updated by the VPB board. In addition, when the train stopped, ATP sends the odometer sensors testing sequence to the VPB board.

[iTC\_CC\_ATP-SwRS-0552]

ATP与VPB的接口应当遵循文档[REF4]描述。

The interface between ATP and VPB shall be compliant with [REF4] document.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0064], [iTC\_CC\_VLE-2-VPB-2-SyID-0002], [iTC\_CC\_VLE-2-VPB-2-SyID-0013], [iTC\_CC\_VLE-2-VPB-2-SyID-0014], [iTC\_CC\_VLE-2-VPB-2-SyID-0015], [iTC\_CC\_VLE-2-VPB-2-SyID-0016], [iTC\_CC\_VLE-2-VPB-2-SyID-0021], [iTC\_CC\_VLE-2-VPB-2-SyID-0022]

[End]

### Physical level

见文档[REF4]描述。

Refer to [REF4].

### Protocol level

见文档[REF4]描述。

Refer to [REF4].

### Application level

VPB板接收和处理来自里程计和信标天线的信号，其结果由ATP通过寄存器进行读取。VPB板有相对独立的两路通道，分别对应ATP的一个CPU模块，共同组成2取2结构。

VPB board receives and processes the signal from odometer and beacon antenna, and ATP gets the result through the VPB registers. There are two independent channels in the VPB board, and it will correspond to each CPU in ATP respectively to form the structure 2 out of 2.

根据文档[REF4]描述，ATP向VPB板发送信息如Table 4‑5所示。

According to [REF4], the onboard ATP shall send information to VPB as shown in the Table 4‑5.

Table 4‑5 Data send to VPB

|  |  |  |
| --- | --- | --- |
| ID | | Comments |
| CBKRead | |  |
|  | .TestReg | Register for ATP writing sensors test byte |
|  | .ATCkeyReg | Register for ATP writing [ATCkey](#ATCkey) |
|  | .ATCkeyReady | Register for ATP notifying VPB to read [ATCkey](#ATCkey) |
|  | .BMreadReady | Register for ATP notifying VPB that ATP will read beacon message |

**NOTES：**

对于传感器测试字寄存器，是由VLE-2板CPU1的ATP软件写给VPB-2板；而VLE-2板的CPU2的ATP程序，从该寄存器可以读到CPU1写入的测试字。详见文档[REF4]。

For the CPU1 for the VLE-2 board, ATP software writes the sensors testing sequence to the VPB register; And for the CPU2, ATP software read the sensors testing from the VPB register.

根据文档[REF4]描述，ATP通过寄存器从VPB板获取的信息如Table 4‑6所示。

According to the [REF4], the onboard ATP read information from VPB as shown in Table 4‑6.

Table 4‑6 List of data from VPB

|  |  |  |
| --- | --- | --- |
| ID | | Comments |
| CBKWrite | |  |
|  | .SensorReg | Register for C1/2/3/4 sensors status |
|  | .CogCounterReg | Register for odometer cog counter and cog code |
|  | .CalibrationReg | Register for VPB locking cog counter when beacon antenna sent top-loc signal |
|  | .BeaconMsgReg | Register for VPB storing 29 bytes beacon message |
|  | .StatusReg | Register for VPB notifying ATP that the beacon message available |
|  | .RadarReg | Register for speed measurement by doppler radar |

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with CC Non Vital

### Role of interface

ATP和运行在DVCOM板上的CCNV程序共同完成了车载控制器的核心计算功能。ATP通过该接口将已组帧的安全消息发送给冗余ATP、VIOM以及外部系统如ZC、LC、CI，同时也接收来自上述设备的消息。此外，ATP还需将自身的运行状态发送给CCNV供维护诊断使用，并接收CCNV的运行信息，作为实现某些功能的依据。

ATP and CCNV, which run on the DVCOM-2 board, accomplish the core computation functions of the Carborne Controller. Through this interface, ATP send vital messages to the redundant ATP, VIOM as well as external systems such as the [ZC](#ZC), LC, CI and also received messages from these devices. In addition, ATP needs to send its running status to the CCNV for maintenance and receive CCNV's operational commands to achieve certain functions.

[iTC\_CC\_ATP-SwRS-0553]

ATP与CCNV的接口应当遵循文档[REF5]描述。

The interface between ATP and CCNV shall be compliant with [REF5] document.

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0067], [iTC\_CC-SyAD-0068], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0001], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0002], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0005], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0006], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0007], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0008], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0009], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0010], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0011], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0067], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0068], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0069], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0070], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0073], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0074], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0075], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0076], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0077]

[End]

### Physical level

Refer to [REF5].

### Protocol level

Refer to [REF5].

### Application level

#### ATP->CCNV: initialization report

在上电后，ATP软件需要按照文档[REF5]定义的结构，将读取来自Dataplug中存储的列车编号、类型和网络地址，以及来自VLE板上的RTC时钟信息，将其发送给CCNV，如Table 4‑7所示。

After power up, ATP software needs to send train id, train type and network address to the CCNV, which read from the CC data plug according to the [REF5], and the RTC clock information read from VLE-2 board. As shown in Table 4‑7.

Table 4‑7 Content of CCNV initialization report

|  |  |
| --- | --- |
| ID | Comments |
| CcSSID | The sub-system id of the train |
| CcTrainType | Train type |
| CcIpBlue | CC external IP address in the blue network |
| CcIpRed | CC external IP address in the red network |
| CCNVIpBlue | CCNV's internal IP address in the blue network |
| CCNVIpRed | CCNV's internal IP address in the red network |
| VLE\_RTCtime | The RTC timer of VLE board |
| [MemLocation](#MemLocation) | Memorized train location |

#### CCNV->ATP: Non vital request

对于部分非SIL4级的功能，以及外部信息，由CCNV在每周期处理后传给ATP。这些变量的值并不影响ATP的安全性，因此可作为ATP计算的输入或判断的条件。

For some part of functions that does not regard as SIL4, CCNV shall handle them and transmit the request to ATP. These variables will not affect the safety of ATP, so it can be used as input or judging criteria for ATP calculation.

[iTC\_CC\_ATP-SwRS-0473]

NonVitalRequest，ATP软件每周期查询是否有来自CCNV的非安全消息：

* 如果没有CCNV消息，则认为本周期未收到该消息；
* 如果收到CCNV消息，则按照[REF5]定义进行CRC校验：
* 如果CRC校验正确，则生成如Table 4‑8结构的全局变量；
* 如果CRC校验错误，则认为本周期未收到该消息。

Every cycle, ATP software queries whether there are non-vital message transmitted from CCNV, and ATP shall receive and store them into [NonVitalRequest](#NonVitalRequest):

* If there is no non-vital message from CCNV, then do nothing;
* Otherwise, ATP shall check the CRC according to [REF5]:
* If the CRC is correct, then ATP generate the structure of global variable such as Table 4‑8;
* Otherwise, ATP shall discard this message.

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0068], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0035]

[End]

Table 4‑8 ATP inputs from CCNV

|  |  |  |
| --- | --- | --- |
| ID | | Comments |
| .SelectedFrontEnd | | CCNV selected active train front |
| .OdometerRef1SpeedUnderThreshold | | Reference speed 1 determines train stop |
| .OdometerRef1Available | | Reference speed 1 available |
| .OdometerRef2SpeedUnderThreshold | | Reference speed 2 determines train stop |
| .OdometerRef2Available | | Reference speed 2 available |
| .EmergencyBrakingNotRequested | | CCNV does not request to trigger EB |
| .VitalParkingBrakingNotRequested | | CCNV does not request to trigger PB |
| .MasterCCcore | | The CC is master or not |
| .RouteSetNotNeeded | | CCNV determines route set is not needed |
| .TrainInCorrectlyDockedZone | | CCNV determines train has correctly docked |
| .PSDoperation\_A | | CCNV order for PSD on side A |
| .PSDoperation\_B | | CCNV order for PSD on side B |
| .NTPtime | | NTP time CCNV used |
| <REPEAT> × 4 ※ | |  |
|  | .PSB1NotDisabled | CCNV enable PSB1 output |
|  | .PSB2NotDisabled | CCNV enable PSB1 output |
|  | .DSB1NotDisabled | CCNV enable DSB1 output |
|  | .DSB2NotDisabled | CCNV enable DSB1 output |
|  | .DSB3NotDisabled | CCNV enable DSB1 output |
|  | .DSB4NotDisabled | CCNV enable DSB1 output |
|  | .DSB5NotDisabled | CCNV enable DSB1 output |
|  | .DSB6NotDisabled | CCNV enable DSB1 output |
|  | .DSB7NotDisabled | CCNV enable DSB1 output |
|  | .DSB8NotDisabled | CCNV enable DSB1 output |
| <REPEAT> × 40 | |  |
|  | .NonVitalIntermediateData[1…40] | Project configurable no vital intermediate data |
| .CancelSignal | | Cancel signal at CBTC mode |
| .OverlapRelease | | Overlap release request at block mode |
| .VariantRequestCbiId | | The first CBI ATP need to communicate on block mode |
| .reserved | | reserved |

※ 其中四个VIOM的传输顺序依次是位于车头1的VIOM1-1，VIOM1-2，车头2的VIOM2-1，VIOM2-2。

The sequence of the four VIOMs in transportation shall be, in turn, VIOM1-1 and VIOM1-2 in CAB1, and VIOM2-1 and VIOM2-2 in CAB2.

#### ATP->CCNV: Non vital report

ATP需要将每周期计算的时间、状态、定位、输出命令等信息发送给CCNV，CCNV将根据这些信息执行自己的控制功能，或者将其转发给DMI等设备进行显示。

ATP needs to send the information to CCNV including cycle time, internal status, location and vital output order etc. Based on this information, CCNV executes the control functions or forward this information to DMI or other devices.

[iTC\_CC\_ATP-SwRS-0447]

NonVitalReport，结构如Table 4‑9所示，表示每周期ATP输出给CCNV的非安全信息。

* 初始化时，设置为全限制状态
* 正常运行时，按照Table 4‑9所示设置ATP输出信息：

Every cycle, ATP shall send the non-vital outputs [NonVitalReport](#NonVitalReport) to CCNV, whose structure is shown in Table 4‑9. The rules are as follows:

* At initialization, set all these information at restrictive state;
* In normal operation, set the ATP status in accordance with the Table 4‑9

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0068], [iTC\_CC-SyAD-0194], [iTC\_CC-SyAD-0200], [iTC\_CC-SyAD-0223], [iTC\_CC-SyAD-0224], [iTC\_CC-SyAD-0247], [iTC\_CC-SyAD-0337], [iTC\_CC-SyAD-0360], [iTC\_CC-SyAD-0407], [iTC\_CC-SyAD-0408], [iTC\_CC-SyAD-0409], [iTC\_CC-SyAD-0760], [iTC\_CC-SyAD-0411], [iTC\_CC-SyAD-0410], [iTC\_CC-SyAD-0137], [iTC\_CC-SyAD-0170], [iTC\_CC-SyAD-0271], [iTC\_CC-SyAD-0274], [iTC\_CC-SyAD-0318], [iTC\_CC-SyAD-0329], [iTC\_CC-SyAD-0350], [iTC\_CC-SyAD-0351], [iTC\_CC-SyAD-0361], [iTC\_CC-SyAD-0364]

[End]

Table 4‑9 Structure of ATP non-vital report

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | ATP outputs |
| ATP status related | | | |
|  | .LoopHour | | [ATPtime](#ATPtime)(k) |
|  | .LatestTimeOtherCore | | [OtherATPminTime](#OtherATPminTime)(k) |
|  | .RtcTime | | [RTCtime](#RTCtime)(k) |
|  | .BlockModeEOAAvailable | | [BlockModeEOAvalid](#BlockModeEOAvalid)(k) |
|  | .TractionAuthorisedSenseEnd\_1 | | [TractionAuthorisedSenseEnd1](#TractionAuthorisedSenseEnd1)(k) |
|  | .TractionAuthorisedSenseEnd\_2 | | [TractionAuthorisedSenseEnd2](#TractionAuthorisedSenseEnd2)(k) |
|  | .OutOfCode | | [OutOfCode](#OutOfCode)(k) |
|  | .SafeTimeFailed | | [SafeTimerFailed](#SafeTimerFailed)(k) |
|  | .TrainFrontEnd | | [TrainFrontEnd](#TrainFrontEnd)(k) |
| Beacon information | | | |
|  | .BeaconCount | | [BeaconCount](#BeaconCount)(k) |
|  | .BeaconId | | [BeaconMessage](#BeaconMessage)(k).ID |
|  | .BlockModeVariants | | [BeaconMessage](#BeaconMessage)(k).Variants |
|  | .BMBeaconDefaultMessage | | [BeaconMessage](#BeaconMessage)(k).[DefaultMessage](#DefaultMessage) |
|  | .BMBeaconVariantAvailable | | [BeaconMessage](#BeaconMessage)(k).[BlockModeVariantAvailable](#BlockModeVariantAvailable) |
| Doors control information | | | |
|  | .DoorOpeningEnabledSide\_A | | [EnableDoorOpening\_A](#EnableDoorOpening_A)(k) |
|  | .DoorOpeningEnabledSide\_B | | [EnableDoorOpening\_B](#EnableDoorOpening_B)(k) |
|  | .HoldDoorClosedOnSide\_A | | [HoldDoorsClosed\_A](#HoldDoorsClosed_A)(k) |
|  | .HoldDoorClosedOnSide\_B | | [HoldDoorsClosed\_B](#HoldDoorsClosed_B)(k) |
|  | .HoldDoorClosedOnEnd\_1 | | [EmergencyDetrainDoorLockingEnd1](#EmergencyDetrainDoorLockingEnd1)(k) |
|  | .HoldDoorClosedOnEnd\_2 | | [EmergencyDetrainDoorLockingEnd2](#EmergencyDetrainDoorLockingEnd2)(k) |
|  | .PSDStateSide\_A | | [PSDstatusNonVital\_A](#PSDstatusNonVital_A)(k) |
|  | .PSDStateSide\_B | | [PSDstatusNonVital\_B](#PSDstatusNonVital_B)(k) |
| Train kinematics information | | | |
|  | .FilteredStop | | [TrainFilteredStopped](#TrainFilteredStopped)(k) |
|  | .KinematicsValid | | [ValidTrainKinematic](#ValidTrainKinematic)(k) |
|  | .OdometerValid | | [OdometerState](#OdometerState)(k)==**INITIALIZED** |
|  | .PermanentLockedAxle | | [UnrecoverableAxleLocked](#UnrecoverableAxleLocked)(k) |
|  | .OdometerAxleMotorized | | [OdometerAxleMotorized](#OdometerAxleMotorized)(k) |
|  | .MaxCalibration | | [MaxCogCalibration](#MaxCogCalibration)(k) |
|  | .MinCalibration | | [MinCogCalibration](#MinCogCalibration)(k) |
|  | .OdometerState | | [OdometerState](#OdometerState)(k) |
|  | .SlidingState | | [MotionOverEstimationState](#MotionOverEstimationState)(k) |
|  | .SlippingState | | [MotionUnderEstimationState](#MotionUnderEstimationState)(k) |
|  | .ReverseSpeedRestriction | | [ReverseSpeedRestrictions](#ReverseSpeedRestrictions)(k) |
|  | .RadarSpeedValid | | [RadarSpeedValid](#RadarSpeedValid)(k) |
|  | .RadarSpeedDirection | | [RadarDirection](#RadarDirection)(k) |
|  | .RadarSpeed | | [RadarRawSpeed](#RadarRawSpeed)(k) |
| Train localization information | | | |
|  | .RouteExclusivityNotGuaranted | | [HazardousMotionOnNonExclusiveRoute](#HazardousMotionOnNonExclusiveRoute)(k) |
|  | .TrainLocated | | [TrainLocalized](#TrainLocalized)(k) |
|  | .TrainPermanentFailureForLocationFault | | [LocPermanentFailure](#LocPermanentFailure)(k) |
|  | .TrainLocatedOnKnownPath | | [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k) |
|  | .TrainAuthorizationForSweepping | | [LocalizedAuthorizationForSweepping](#LocalizedAuthorizationForSweepping)(k) |
|  | .MemorizedLocationNotConfirmed | | [MemLocationNotConfirmed](#MemLocationNotConfirmed)(k) |
|  | .TrainLocalizationState | | [LocalizationState](#LocalizationState)(k) |
|  | .TrainEnd2Orientation | | [TrainLocation](#TrainLocation)(k).Ext2.Ort |
|  | .TrainDeltaLocation | | [TrainLocation](#TrainLocation)(k).Uncertainty |
|  | .TrainEnd2BlockId | | [TrainLocation](#TrainLocation)(k).Ext2.Block |
|  | .TrainEnd2Abscissa | | [TrainLocation](#TrainLocation)(k).Ext2.Abscissa |
|  | .TrainDelocatedByCoupled | | [CoupledTypeInconsistent](#CoupledTypeInconsistent)(k) |
|  | .TrainCoupledType | | [TrainCoupledType](#TrainCoupledType)(k) |
| Emergency breaking reasons | | | |
|  | .ParkingBrakingNotRequested | | [InhibitParkingBrake](#InhibitParkingBrake)(k) |
|  | .EmergencyBrakingNotRequested | | [InhibitEmergencyBrake](#InhibitEmergencyBrake)(k) |
|  | .EBforRollbackOverspeed | | [EBforRollbackOverSpeed](#EBforRollbackOverSpeed)(k) |
|  | .EBforReverseOverspeed | | [EBforReverseOverSpeed](#EBforReverseOverSpeed)(k) |
|  | .EBforDepatureWhithoutTDCL | | [EBforDepartureWithoutTDCL](#EBforDepartureWithoutTDCL)(k) |
|  | .EBforOverEnergy | | [EBforOverEnergy](#EBforOverEnergy)(k) |
|  | .EBforEvacuationWithTrainStopped | | [EBforEvacuationWithTrainStopped](#EBforEvacuationWithTrainStopped)(k) |
|  | .EBforEvacuationWhileTrainLeavingStation | | [EBforEvacuationWhileTrainLeavingStation](#EBforEvacuationWhileTrainLeavingStation)(k) |
|  | .EBforUnexpectedPsdOpening | | [EBforUnexpectedPSDopening](#EBforUnexpectedPSDopening)(k) |
|  | .EBforMoralTime | | [EBonNonExclusiveRoute](#EBonNonExclusiveRoute)(k) |
|  | .EBforRMOverspeed | | [EBforRMoverSpeed](#EBforRMoverSpeed)(k) |
|  | .EBforUndectableDangerRisk | | [EBforUndetectableDangerRisk](#EBforUndetectableDangerRisk)(k) |
|  | .EBforApproachableSignalOverrun | | [ApproachableSignalOverrun](#ApproachableSignalOverrun)(k) |
|  | .EBforPbNotAppliedDueTo\_PSD | | [EBforPBnotAppliedDueToPSD](#EBforPBnotAppliedDueToPSD)(k) |
|  | .EBforPbNotAppliedDueToTrainDoors | | [EBforPBnotAppliedDueToTrainDoors](#EBforPBnotAppliedDueToTrainDoors)(k) |
|  | .EBforMemorizedLocationOverSpeed | | [EBforMemorizedLocationOverSpeed](#EBforMemorizedLocationOverSpeed)(k) |
|  | .EBforNotAllTrainEndHoldDoorsClosed | | [EBforNotAllTrainEndHoldDoorsClosed](#EBforNotAllTrainEndHoldDoorsClosed)(k) |
|  | .EBforUnrecoverableRollbackOverspeed | | [UnrecoverableRollbackOverSpeed](#UnrecoverableRollbackOverSpeed)(k) |
|  | .EBforIncompatibleDistantATP | | [IncompatibleDistantATP](#IncompatibleDistantATP)(k) |
|  | .EBforCcWorkOvertime | | [CCworkOvertime](#CCworkOvertime)(k) |
|  | .EBforMovingWithoutTDCL | | [EBforMovingWithoutTDCL](#EBforMovingWithoutTDCL)(k) |
| LC version report | | | |
|  | .VersionReport | | [VersionFromCCreport](#VersionFromCCreport)(k) |
| No danger for RM selectable information | | | |
|  | .NoDangerForVitalSpeedLimitExceed | | not [TrainPossiblyInOverEnergy](#TrainPossiblyInOverEnergy)(k) |
|  | .NoDangerForRollbackOverspeed | | not [RollbackOverSpeed](#RollbackOverSpeed)(k) |
|  | .NoDangerForReverseOverspeed | | not [ReverseOverSpeed](#ReverseOverSpeed)(k) |
|  | .NoDangerForRmOverspeed | | [NoDangerForRMoverSpeed](#NoDangerForRMoverSpeed)(k) |
|  | .NoDangerForMemorizedLocationOverspeed | | [NoDangerforMemorizedLocationOverSpeed](#NoDangerforMemorizedLocationOverSpeed)(k) |
|  | .NoDangerForDepartureWithoutAllDoorsClosed | | [NoDangerForDepartureWithoutTDCL](#NoDangerForDepartureWithoutTDCL)(k) |
|  | .NoDangerForUnexpectedPsdOpening | | [NoDangerForUnexpectedPSDopening](#NoDangerForUnexpectedPSDopening)(k) |
|  | .NoDangerForPbNotAppliedDueToTrainDoors | | [NoDangerForTrainDoorsNotClosedAndLocked](file:///D:\ATP\3-Requirement\ATP-SwRS.docx#NoDangerForTrainDoorsNotClosedAndLocked)(k) |
|  | .NoDangerForPbNotAppliedDueToPsd | | [NoDangerForPSDnotClosedAndLocked](file:///D:\ATP\3-Requirement\ATP-SwRS.docx#NoDangerForPSDnotClosedAndLocked)(k) |
|  | .NoDangerForHoldDoorsClosedTrainEnd1 | | [HoldDoorsClosedTrainEnd1](#HoldDoorsClosedTrainEnd1)(k) |
|  | .NoDangerForHoldDoorsClosedTrainEnd2 | | [HoldDoorsClosedTrainEnd2](#HoldDoorsClosedTrainEnd2)(k) |
|  | .NoDangerForEvacuationWithTrainStopped | | not [EvacuationWithTrainStopped](#EvacuationWithTrainStopped)(k) |
|  | .NoDangerForEvacuationWhileLeavingStation | | not [EvacuationWhileLeavingStation](#EvacuationWhileLeavingStation)(k) |
|  | .NoDangerForUndertectableDangerRisk | | not [UndetectableDangerRiskForNoNUDE](#UndetectableDangerRiskForNoNUDE)  and not [UndetectDangerMotionWithoutNUDE](#UndetectDangerMotionWithoutNUDE) |
|  | .NoDangerForNonExlcusiveRouteOnEnd1 | | [RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1)(k)  or ([TrainFrontEnd](#TrainFrontEnd)(k) == **END\_2**) |
|  | .NoDangerForNonExlcusiveRouteOnEnd2 | | [RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2)(k)  or ([TrainFrontEnd](#TrainFrontEnd)(k) == **END\_1**) |
|  | .NoDangerForMovingWithoutTDCL | | [NoDangerForMovingWithoutTDCL](#NoDangerForMovingWithoutTDCL)(k) |
| Applied LC messages loop hour | | | |
|  | .ATPAppliedTsrLoopHour[1…16] | | [TSRdownloadContent](#TSRdownloadContent)[1…16].CcLoopHour |
|  | | .ATPAppliedVersionLoopHour[1…16] | [VersionAuthorization](#VersionAuthorization)[1…16].CcLoopHour |
| Communicated with CBI information | | | |
|  | .CbiVariantUsed | | [CBIvariantMoreAvailableThanBeacon](#CBIvariantMoreAvailableThanBeacon)[1…32](k) |
|  | .AppliedCbiMessageLoopHour[1…32] | | [AppliedCBIvariantLoopHour](#AppliedCBIvariantLoopHour)[1…32](k) |
|  | .ReceivedCbiVariantLowValidity[1…32] | | [CBIvariantLowValidity](#CBIvariantLowValidity)[1…32](k) |
| Logical inputs | | | |
|  | .PermissiveZoneLogicalInput[1…4] | | [PermissiveZoneLogicalInput](#PermissiveZoneLogicalInput)[1…4](k) |
|  | .NotRestrictiveZoneLogicalInput[1…4] | | [NotRestrictiveZoneLogicalInput](#NotRestrictiveZoneLogicalInput)[1…4](k) |
| Vital inputs | | | |
|  | .TrainUnitIntegrity | | [TrainUnitIntegrity](#TrainUnitIntegrity)(k) |
|  | .TrainNotCoupledByEnd\_1 | | not [CoupledByEnd1](#CoupledByEnd1)(k) |
|  | .TrainNotCoupledByEnd\_2 | | not [CoupledByEnd2](#CoupledByEnd2)(k) |
|  | .TrainMemorisedLocationAuthorised | | [MemorizedLocationAuthorized](#MemorizedLocationAuthorized)(k) |
|  | .DriverInCab\_1 | | [DriverInCab\_1](#DriverInCab_1)(k) |
|  | .DriverInCab\_2 | | [DriverInCab\_2](#DriverInCab_2)(k) |
|  | .NoUndetectableDangerEnd\_1 | | [NoUndetectableDanger\_1](#NoUndetectableDanger_1)(k) |
|  | .NoUndetectableDangerEnd\_2 | | [NoUndetectableDanger\_2](#NoUndetectableDanger_2)(k) |
|  | .TsrControlInhibition | | [TSRcontrolInhibition](#TSRcontrolInhibition)(k) |
|  | .RmrDrivingModeSelected | | [RMRselectedDrivingMode](#RMRselectedDrivingMode)(k) |
|  | .BmOverlapReleasableSendable | | [BMoverlapReleasableSendable](#BMoverlapReleasableSendable)(k) |
|  | .RouteSetNotNeededSendable | | [RouteSetNotNeededSendable](#RouteSetNotNeededSendable)(k) |
|  | .BlockModeVariantValidWhileTemporallyValid | | [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)(k) |
|  | .BlockModeUsed | | [BlockModeUsed](#BlockModeUsed)(k) |
|  | .TrainDoorClosedAndLocked | | [AllTrainDoorsClosedAndLocked](#AllTrainDoorsClosedAndLocked)(k) |
|  | .PsdNotClosedOnTrainStartControlInhibition | | [InhibitControlPSDstatus](#InhibitControlPSDstatus)(k) |
|  | .PassengerEmergencyHandleNotPulledForSides | | [EmergencyHandleNotPulledSide](#EmergencyHandleNotPulledSide)(k) |
|  | .PassengerEmergencyHandleNotPulledForEnd\_1 | | [EmergencyHandleNotPulledEnd1](#EmergencyHandleNotPulledEnd1)(k) |
|  | .PassengerEmergencyHandleNotPulledForEnd\_2 | | [EmergencyHandleNotPulledEnd2](#EmergencyHandleNotPulledEnd2)(k) |
|  | .TrainEmergencyBrakApplied | | [TrainEmergencyBrakeApplied](#TrainEmergencyBrakeApplied)(k) |
|  | .TrainParkingBrakeApplied | | [TrainParkingBrakeApplied](#TrainParkingBrakeApplied)(k) |
|  | .MotionProtectionInhibition | | [MotionProtectionInhibition](#MotionProtectionInhibition)(k) |
|  | .SignalOverrideSendable | | [SignalOverrideSendable](#SignalOverrideSendable)(k) |
|  | .PsdOpeningProtectionInhibition | | [InhibitPSDopeningSupervisedByATP](#InhibitPSDopeningSupervisedByATP)(k) |
|  | .EvacuationWithTrainStoppedProtectionInhibition | | [InhibitProtectionEvacuationWithStop](#InhibitProtectionEvacuationWithStop)(k) |
|  | .EvacuationInSanctionDistanceProtectionInhibition | | [InhibitProtectionEvacuationInDistance](#InhibitProtectionEvacuationInDistance)(k) |
|  | .TdclOnTrainStartControlInhibition | | [InhibitControlTrainDoorsStatus](#InhibitControlTrainDoorsStatus)(k) |
|  | .AtcControlTrain | | [ATCcontrolledTrain](#ATCcontrolledTrain)(k) |
|  | .MovingWithoutTDCLprotectionInhibition | | [InhibitProtectionMovingWithoutTDCL](#InhibitProtectionMovingWithoutTDCL)(k) |
|  | .ConditionForRmLimitSpeed[1…7] | | [ConditionForRMlimitSpeed](#ConditionForRMlimitSpeed)[1…7](k) |
|  | .CoercedPermissive[1…4] | | [CoercedPermissive](#CoercedPermissive)\_1~4 (k) |
|  | .NotCoercedRestrictive[1…4] | | [NotCoercedRestrictive](#NotCoercedRestrictive)\_1~4](k) |
| <End> | | | |

[iTC\_CC\_ATP-SwRS-0481]

ATP软件每周期将[NonVitalReport](#NonVitalReport)信息发送给CCNV。

Every cycle, ATP shall send [NonVitalReport](#NonVitalReport) message to CCNV

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0068], [iTC\_CC-SyAD-0407], [iTC\_CC-SyAD-0408], [iTC\_CC-SyAD-0409], [iTC\_CC-SyAD-1004], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0020]

[End]

#### ATP->CCNV maintenance information

[iTC\_CC\_ATP-SwRS-0742]

所有在功能模块“Outputs”部分中定义为“Observable”的需求变量，ATP均将其发送给CCNV，由其转发给维护工具存储或显示。

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0408], [iTC\_CC-SyAD-1418] , [iTC\_CC-SyAD-1423]

[End]

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with Distant ATP

### Role of interface

本接口用于ATP通过DVCOM板接收来自远端ATP的同步信息，并将自身的同步信息通过该接口发送给远端ATP。

[iTC\_CC\_ATP-SwRS-0602]

ATP与远端ATP的接口应当遵循文档[REF6]描述，通信配置来自[REF11]。

The interface between ATP and distant ATP shall be compliant with [REF6] document.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0067], [iTC\_CC-SyAD-0068], [iTC\_CC\_ATP\_SwHA-0243], [iTC\_CC\_ATP-Offline-SyID-0012], [iTC\_CC\_ATP-Offline-SyID-0013]

[End]

### Physical level

见文档[REF6]描述。

### Protocol level

见文档[REF6]描述。

### Application level

#### Local ATP->Distant ATP

[iTC\_CC\_ATP-SwRS-0372]

ATP软件每周期计算要发送给冗余ATP的信息CCsynchroReport，其结构为ST\_SYNCHRO\_REPORT。

* 初始化时，将其设置为全限制状态
* 正常运行时，按照Table 4‑10设置ATP计算的相应变量。

At each cycle, ATP shall send a [CCsynchroReport](#CCsynchroReport), which structure is ST\_SYNCHRO\_REPORT, to the redundant ATP located at the other cab.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0411], [iTC\_CC-SyAD-0963], [iTC\_CC-SyAD-1210]

[End]

根据[REF6]，ATP通过CCNV向冗余ATP发送两端ATP的同步信息，其结构ST\_SYNCHRO\_REPORT如Table 4‑10所示。

According to [REF6], ATP sends the synchronization message to the redundant ATP. The structure of ST\_SYNCHRO\_REPORT shows in Table 4‑10.

Table 4‑10 ATP outputs to redundant ATP

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | ATP outputs |
| ST\_SYNCHRO\_REPORT | | | [CCsynchroReport](#CCsynchroReport) |
|  | | .CurrentTime | [ATPtime](#ATPtime)(k) |
|  | | .LatestTimeOtherCore | [OtherATPminTime](#OtherATPminTime)(k) |
|  | | .CoreId | [CoreId](#CoreId)(k) |
|  | | .BeaconId | [BeaconLastObtained](#BeaconLastObtained)(k).Id |
|  | | .EnableDoorOpening\_A | [LocalATPenableDoorOpening\_A](#LocalATPenableDoorOpening_A)(k) |
|  | | .EnableDoorOpening\_B | [LocalATPenableDoorOpening\_B](#LocalATPenableDoorOpening_B)(k) |
|  | | .PsdManagerOpeningOrder | [PSDplatformManagerOpeningOrder](#PSDplatformManagerOpeningOrder)(k) |
|  | | .PsdIdSide\_A | [PSDzoneStatus\_A](#PSDzoneStatus_A)(k).Id |
|  | | .PsdValiditySide\_A | [PSDzoneStatus\_A](#PSDzoneStatus_A)(k).Validity |
|  | | .PsdClosedSide\_A | [PSDzoneStatus\_A](#PSDzoneStatus_A)(k).AllPSDclosed |
|  | | .PsdIdSide\_B | [PSDzoneStatus\_B](#PSDzoneStatus_B)(k).Id |
|  | | .PsdValiditySide\_B | [PSDzoneStatus\_B](#PSDzoneStatus_B)(k).Validity |
|  | | .PsdClosedSide\_B | [PSDzoneStatus\_B](#PSDzoneStatus_B)(k).AllPSDclosed |
|  | | .ZCversion[1..16] | [TrackMap](#TrackMap).[ZC](#ZC)[1...16].Version |
|  | | .LocatedOnKnownPath | [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k) |
|  | | .LocatedWithMemLocation | [TrainPresumablyLocalized](#TrainPresumablyLocalized)(k) |
|  | | .Location.Ext2 | [TrainLocation](#TrainLocation)(k).Ext2 |
|  | | .Location.Uncertainty | [TrainLocation](#TrainLocation)(k).Uncertainty |
|  | | .Location.Ext1 | [TrainLocation](#TrainLocation)(k).Ext1 |
|  | | .SleepZoneId | [TrainIncludedInSleepingZone](#TrainIncludedInSleepingZone)(k).Id |
|  | | .SleepZoneVersion | [TrainIncludedInSleepingZone](#TrainIncludedInSleepingZone)(k).Version |
|  | | .MotionSinceLastReloc | [MotionSinceLastReloc](#MotionSinceLastReloc)(k) |
|  | | .MotionSinceMemLoc | [MotionSinceMemorizedLocation](#MotionSinceMemorizedLocation)(k) |
|  | | .TrainFilteredStopped | [TrainFilteredStopped](#TrainFilteredStopped)(k) |
|  | | .SafetyParameterVersion | [ATPsetting](#ATPsetting).SafetyParameterVersion |
|  | | .SafetyApplicationVersion | [SafeApplicationVersion](#SafeApplicationVersion)(k) |
|  | | .CC\_SSID | [SubSystemId](#SubSystemId)(k) |
|  | | .OverlapExpired | [OverlapTimer](#OverlapTimer)(k) is **0** |
|  | VitalChecksum\_1 | | SACEM vital checksum 1 |
|  | VitalChecksum\_2 | | SACEM vital checksum 2 |

[iTC\_CC\_ATP-SwRS-0475]

ATP软件每周期将同步信息[IdenticalCCsyncReport](#IdenticalCCsyncReport)按照[REF6] 定义格式，以SACEM安全通信协议发送给CCNV，由其转发给冗余ATP。

Every cycle, ATP software sends [IdenticalCCsyncReport](#IdenticalCCsyncReport) message to CCNV according to [REF6] as the SACEM communication protocol. The CCNV will forward this message to the redundant ATP.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0068], [iTC\_CC-SyAD-0961], [iTC\_CC\_ATP\_SwHA-0012], [iTC\_CC\_ATP\_SwHA-0161], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0027], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0028]

[End]

#### Distant ATP -> Local ATP

根据[REF6]，ATP通过CCNV获取来自冗余ATP的同步信息[CCNV\_RedundantATPmessage](#CCNV_RedundantATPmessage)，其结构ST\_SYNCHRO\_REPORT如Table 4‑10所示。

According to [REF6], ATP software should receive the synchronization report from the redundant ATP, as shown in Table 4‑10.

[iTC\_CC\_ATP-SwRS-0469]

OtherCCsynchroReport，ATP软件每周期查询CCNV是否有转发来自冗余ATP的安全消息[CCNV\_RedundantATPmessage](#CCNV_RedundantATPmessage)，据此解析生成[OtherCCsynchroReport](#OtherCCsynchroReport)消息：

* 如果没有冗余ATP消息，或者CRC校验或SACEM校核字错误，则认为本周期未收到上述消息；
* 否则，生成如ST\_SYNCHRO\_REPORT结构的全局变量。

Every cycle, ATP software queries whether there is redundant ATP message [CCNV\_RedundantATPmessage](#CCNV_RedundantATPmessage) transmitted from CCNV. ATP shall receive and store the message into [OtherCCsynchroReport](#OtherCCsynchroReport). The receiving and storing rules are following:

* If there is no message from the redundant ATP, or the CRC or vital checksum of this message checked failure, ATP shall discard this message;
* Otherwise, ATP shall generate the structure of global variable such as ST\_SYNCHRO\_REPORT.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0961], [iTC\_CC-SyAD-0963], [iTC\_CC-SyAD-0964], [iTC\_CC\_ATP\_SwHA-0012], [iTC\_CC\_ATP\_SwHA-0161], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0046], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0047]

[End]

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with VIOM

### Role of interface

本接口用于ATP通过DVCOM板接收来自VIOM采集的车辆输入信息，并将计算得到的车辆控制命令通过该接口发送给VIOM。

[iTC\_CC\_ATP-SwRS-0603]

ATP与VIOM的接口应当遵循文档[REF5]描述，通信配置来自[REF11]。

The interface between ATP and VIOM shall be compliant with [REF5] document.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0067], [iTC\_CC-SyAD-0068], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0001], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0002], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0005], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0006], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0007], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0008], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0009], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0010], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0067], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0068], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0070], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0060], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0011], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0074], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0072], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0016], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0075], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0076], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0077] , [iTC\_CC\_ATP-Offline-SyID-0012], [iTC\_CC\_ATP-Offline-SyID-0013]

[End]

### Physical level

见文档[REF5]描述。

### Protocol level

见文档[REF5]描述。

### Application level

#### VIOM->ATP

根据[REF5]，ATP通过CCNV接收来自两端驾驶室共4个VIOM的安全输入端口状态。对于同一端的2个VIOM，呈冗余关系，只要二者间有一个采到某路输入为真，则ATP就认为该输入为真。对于两端驾驶室的VIOM1VitalInput和VIOM2VitalInput，其内容如Table 4‑11所示。

According to [REF5], the ATP software received from both ends of the train for four VIOM vital inputs message. The vital inputs of the cab, are the result of “logical OR” operation between the two VIOM’s of the same cab. For each VIOM there are as shown in Table 4‑11.

Table 4‑11 ATP inputs from VIOM

|  |  |  |
| --- | --- | --- |
| ID | | Comments |
| ST\_VIOM\_INPUT | | 来自VIOM的安全输入信息 |
|  | .VDI[**MAX\_VITAL\_INPUT\_NB**] | VIOM采集的离散安全输入 |
|  | .ViomLoopHour | VIOM的loopHour |
|  | .AtpLoopHour | VIOM之前收到的ATP消息中所带的时间信息 |
|  | .VitalChecksum1 | SACEM vital checksum 1 |
|  | .VitalChecksum2 | SACEM vital checksum 2 |

#### ATP->VIOM:

根据[REF5]，ATP通过CCNV向分别向列车两端共4个VIOM发送安全输出信息，其消息结构如Table 4‑12所示。

According to [REF5], ATP send vital outputs to the four VIOM respectively and the message structure as shown in Table 4‑12.

[iTC\_CC\_ATP-SwRS-0474]

ATP软件每周期将包含上下模块输出的车辆控制命令[IdenticalVIOM1Out](#IdenticalVIOM1Out)和[IdenticalVIOM2Out](#IdenticalVIOM2Out)按照[REF5]定义格式，以SACEM安全通信协议发送给CCNV，由其转发给两端车头的4个VIOM。

Every cycle, ATP Software sends the rolling stock commands [IdenticalVIOM1Out](#IdenticalVIOM1Out) and [IdenticalVIOM2Out](#IdenticalVIOM2Out) to CCNV according to [REF5] as the SACEM communication protocol. The CCNV will forward these messages to 4 VIOMs of both train ends

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0031], [iTC\_CC-SyAD-0114], [iTC\_CC-SyAD-0961], [iTC\_CC\_ATP\_SwHA-0017], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0021], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0022]

[End]

Table 4‑12 Structure of ATP outputs to VIOM

|  |  |  |
| --- | --- | --- |
| ID | | Comments |
| ST\_VIOM\_OUT | |  |
|  | *.VDO\_UP[***MAX\_VITAL\_OUTPUT\_NB***]* | CPU1计算的安全输出  Vital outputs of CPU1's ATP |
|  | *.VDO\_DOWN[***MAX\_VITAL\_OUTPUT\_NB***]* | CPU2计算的安全输出  Vital outputs of CPU2's ATP |
|  | .AtpLoopHour | 当前ATP的loopHour号  ATP's loop hour |
|  | ViomLoopHour | 收到的VIOM的loopHour  VIOM's loop hour |
|  | .TraceUp | CPU1的安全时钟校核字  Checkword for two CPUs' synchronization of CPU1 |
|  | .DtUp | CPU1的动态时间  Dynamic time of this ATP cycle of CPU1 |
|  | .TraceDown | CPU2的安全时钟校核字  Checkword for two CPUs' synchronization of CPU2 |
|  | .DtDown | CPU2的动态时间  Dynamic time of this ATP cycle of CPU2 |
|  | .SafeTimeCheckedUp | CPU1的硬件安全时钟工作正常  Checkword for VLE safe clock of CPU1 |
|  | .SafeTimeCheckedDown | CPU2的硬件安全时钟工作正常  Checkword for VLE safe clock of CPU2 |
|  | .MasterCore | 是否是主控CC  Whether the ATP is the master core or not |
|  | VitalChecksum1 | SACEM安全校核字1  Vital checksum 1 |
|  | VitalChecksum2 | SACEM安全校核字2  Vital checksum 2 |

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with ZC

### Role of interface

本接口用于ATP通过DVCOM板接收来自ZC的EOA和变量消息，并将ATP计算得到的位置报文通过该接口发送给ZC。

[iTC\_CC\_ATP-SwRS-0604]

ATP与ZC的接口应当遵循文档[REF7]描述，通信配置来自[REF11]。

The interface between ATP and ZC shall be compliant with [REF7] document.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0067], [iTC\_CC-SyAD-0068], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0001], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0002], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0005], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0006], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0007], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0008], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0009], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0010], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0067], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0068], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0070], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0011], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0074], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0075], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0076], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0077] , [iTC\_CC\_ATP-Offline-SyID-0012], [iTC\_CC\_ATP-Offline-SyID-0013]

[End]

### Physical level

见文档[REF7]描述。

### Protocol level

见文档[REF7]描述。

### Application level

#### ATP->ZC: Location report

根据[REF7]，ATP通过CCNV向ZC发送列车位置信息。

[iTC\_CC\_ATP-SwRS-0466]

LocReport，ATP每周期需计算发送给ZC的信息，如Table 4‑13所示。

Every cycle, ATP shall generate the Location Report message to the [ZC](#ZC), as shown in Table 4‑13.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0411], [iTC\_CC-SyAD-1217], [iTC\_CC-SyAD-0221], [iTC\_CC\_ATP\_SwHA-0157]

[End]

Table 4‑13 ATP outputs to [ZC](#ZC)

|  |  |  |
| --- | --- | --- |
| ID | | Comments |
| ST\_LOC\_REPORT | | LocReport |
|  | .TrainId | [SubSystemId](#SubSystemId)(k) |
|  | .TrainUnitHeadCabId | [TrainFrontEnd](#TrainFrontEnd)(k) |
|  | .TrainUnitHeadCabOrientation | [TrainHeadOrientation](#TrainHeadOrientation)(k) |
|  | .TrainUnitHeadCabBlockId | [TrainHeadMinLocation](#TrainHeadMinLocation)(k).Block |
|  | .TrainUnitHeadCabMinAbscissa | [TrainHeadMinLocation](#TrainHeadMinLocation)(k).Abscissa |
|  | .TrainUnitHeadCabCoupledStatus | [TrainHeadCoupledStatus](#TrainHeadCoupledStatus)(k) |
|  | .TrainUnitTailCabId | [TrainTailCabId](#TrainTailCabId)(k) |
|  | .TrainUnitTailCabOrientation | [TrainTailOrientation](#TrainTailOrientation)(k) |
|  | .TrainUnitTailCabBlockId | [TrainTailMinLocation](#TrainTailMinLocation)(k).Block |
|  | .TrainUnitTailCabMinAbscissa | [TrainTailMinLocation](#TrainTailMinLocation)(k).Abscissa |
|  | .TrainUnitTailCabCoupledStatus | [TrainTailCoupledStatus](#TrainTailCoupledStatus)(k) |
|  | .LocationError | [LocationError](#LocationError)(k) |
|  | .LocalizedStatus | [TrainCoupledType](#TrainCoupledType)(k) is **TRAIN\_NO\_COUPLED**  and [LocalizedAuthorizationForSweepping](#LocalizedAuthorizationForSweepping)(k) |
|  | .TrainUnitConfirmedLocalized | [TrainCoupledType](#TrainCoupledType)(k) is **TRAIN\_NO\_COUPLED**  and [TrainConfirmedLocalized](#TrainConfirmedLocalized)(k) |
|  | .TrainUnitImmobilityStatus | [TrainFilteredStopped](#TrainFilteredStopped)(k) |
|  | .TrainUnitRouteSetNotNeeded | [TrainRouteSetNotNeeded](#TrainRouteSetNotNeeded)(k) |
|  | .TrainUnitCorrectDocking | [TrainCorrectDocking](#TrainCorrectDocking)(k) |
|  | .TrainUnitSpeed | [LocReportSpeed](#LocReportSpeed)(k) |
|  | .TrainUnitMonitoringMode | [TrainMonitoringMode](#TrainMonitoringMode)(k) |
|  | .SignalsOverride | [SignalsOverride](#SignalsOverride)(k) |
|  | .AtcControlledTrain | [ATCcontrolledTrain](#ATCcontrolledTrain)(k) |
|  | .ZcVitalAuthorization[1...16] | [ReceivedVersionMessages](#ReceivedVersionMessages)(k).VitalAuthorization[1...16] |
|  | .Boolean[1…16] | [Offline](#Offline).[GetBooleanForZC](#GetBooleanForZC)(1...16) |
|  | .CcLoopHour | [ATPtime](#ATPtime)(k) |
|  | .LcDate | [GroundTimeReference](#GroundTimeReference)(LcId).Time |
|  | .VitalChecksum\_1 | SACEM vital checksum 1 |
|  | .VitalChecksum\_2 | SACEM vital checksum 2 |

[iTC\_CC\_ATP-SwRS-0476]

如果本周期[SendLocReportOnZCunderTrainHead](#SendLocReportOnZCunderTrainHead)为真，ATP将位置报告信息[IdenticalLocReport](#IdenticalLocReport)按照[REF7] 定义格式，以SACEM安全通信协议发送给CCNV，由其转发给本周期列车车头所在的[ZCidUnderTrainHead](#ZCidUnderTrainHead)。

If the [SendLocReportOnZCunderTrainHead](#SendLocReportOnZCunderTrainHead) in this cycle is **True**, ATP shall sends [IdenticalLocReport](#IdenticalLocReport) message to CCNV according to [REF7] as the SACEM communication protocol. The CCNV will forward this message to the [ZC](#ZC) which id is [ZCidUnderTrainHead](#ZCidUnderTrainHead).

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0961], [iTC\_CC\_ATP\_SwHA-0167], [iTC\_CC-SyAD-1008], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0023], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0024]

[End]

[iTC\_CC\_ATP-SwRS-0477]

如果本周期[SendLocReportOnZCunderTrainTail](#SendLocReportOnZCunderTrainTail)为真，ATP将位置报告信息[IdenticalLocReport](#IdenticalLocReport)按照[REF7] 定义格式，以SACEM安全通信协议发送给CCNV，由其转发给本周期列车车尾所在的[ZCidUnderTrainTail](#ZCidUnderTrainTail)。

If the [SendLocReportOnZCunderTrainTail](#SendLocReportOnZCunderTrainTail)this cycle is **True**, ATP shall sends [IdenticalLocReport](#IdenticalLocReport) message to CCNV according to [REF7] as the SACEM communication protocol. The CCNV will forward this message to the [ZC](#ZC) which id is [ZCidUnderTrainTail](#ZCidUnderTrainTail).

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0961], [iTC\_CC\_ATP\_SwHA-0167], [iTC\_CC-SyAD-1008], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0023], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0024]

[End]

#### ZC->ATP: End of authorization

ZC通过来自辖区CC的位置报文，计算每个CC所在列车所能运行的最远位置，作为该CC的EOA。ATP需接收并判断该EOA是否有效，在CBTC运营模式下，如果没有有效的来自ZC的EOA，则ATP将禁止列车移动。

According to the Location Reports of the CC, the [ZC](#ZC) calculates the farthest authorization location of the train, as the EOA of this CC. ATP shall receive and determine the validity of the EOA. In CBTC mode, if no available EOA from the [ZC](#ZC), the ATP shall prevent train moving.

根据[REF7]，ATP通过CCNV获取来自ZC的EOA信息CCNV\_EOAmessage，其结构ST\_EOA\_REPORT如Table 4‑14所示。

According to [REF7], ATP software should receive the EOA message from the [ZC](#ZC), as shown in Table 4‑14.

Table 4‑14 EOA report from [ZC](#ZC)

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | Comments |
| ST\_EOA\_REPORT | | | EOA message |
|  | | .TrainFrontEnd | EOA授权车头 |
|  | | .Classic.Type | ZC发送的EOA类型为ENUM\_EOA\_ZC\_TYPE |
|  | | .Classic.Location.Block | EOA所在Block编号 |
|  | | .Classic.Location.Abscissa | EOA所在Block的坐标 |
|  | | .Classic.CreationTime | EOA创建时间 |
|  | | .WithoutSpacing.Type | SMI区域内使用的EOA类型 |
|  | | .WithoutSpacing.Location.Block | SMI区域内使用的EOA所在Block编号 |
|  | | .WithoutSpacing.Location.Abscissa | SMI区域内使用的EOA所在Block的坐标 |
|  | | .WithoutSpacing.CreationTime | SMI区域内使用的EOA创建时间 |
|  | | .CcLoopHour | EOA消息中所带CC发送时间 |
|  | | .MessageContainerCreationTime | EOA消息创建时间 |
|  | .VitalChecksum1 | | SACEM vital checksum 1 |
|  | .VitalChecksum2 | | SACEM vital checksum 2 |

[iTC\_CC\_ATP-SwRS-0470]

ATP软件每周期查询CCNV是否有转发来自ZC的安全消息[CCNV\_EOAmessage](#CCNV_EOAmessage)和[CCNV\_VariantMessage](#CCNV_VariantMessage)，据此解析生成EOA\_Report和VariantReport：，以及ZC消息所带的SSIDofZC：

* 如果没有ZC消息或者CRC校验或SACEM校核字错误，则认为本周期未收到上述消息；
* 否则，分别生成如ST\_EOA\_REPORT和ST\_VARIANT\_REPORT结构的全局变量。

Every cycle, ATP software queries whether there are [ZC](#ZC) messages [CCNV\_EOAmessage](#CCNV_EOAmessage) and [CCNV\_VariantMessage](#CCNV_VariantMessage) transmitted from CCNV. ATP shall receive and store the message into [EOA\_Report](#EOA_Report) and [VariantReport](#VariantReport), and get the SSID from these messages to generate SSIDofZC. The receiving and storing rules are following:

* If there is no message from [ZC](#ZC), or the CRC or vital checksum of this message checked failure, ATP shall discard this message;
* Otherwise, ATP shall generate the structure of global variable such as ST\_EOA\_REPORT and ST\_VARIANT\_REPORT.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0153], [iTC\_CC-SyAD-0155], [iTC\_CC-SyAD-0159], [iTC\_CC-SyAD-0961], [iTC\_CC-SyAD-0966], [iTC\_CC\_ATP\_SwHA-0024], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0038], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0039]

[End]

**NOTES**:

考虑到校验变量消息的时间限制，要求在进行工程数据设计时，最多允许1个ZC区域包括2个LineSection的变量。即在ZC的变量消息中，最多包含本ZC的2个LS变量和两端相邻ZC的各1个LS的变量。即收到的变量消息中最多包括4个LineSection的变量。

Considering the time consumption for the checksums calculating of variant reports, the rule that one [ZC](#ZC) can manage at most two line sections shall be followed during the project data preparation. Therefore, in one variant report message, there are maximum four line sections' variants (two for current [ZC](#ZC), and each one for the downstream and the upstream adjacent [ZC](#ZC)) sent from the [ZC](#ZC).

#### [ZC](#ZC)->ATP: Variants

ZC在发送EOA消息时，会将其辖区以及相邻ZC的一段线路的变量信息也发给CC。因此对于ATP，在处理EOA时，还需处理来自ZC的变量信息。

When [ZC](#ZC) sent EOA, includes the variants in this [ZC](#ZC) area and adjacent [ZC](#ZC) to CC. Therefore, when received a valid EOA message, ATP shall parse these variants information at the same time.

根据[REF7]，ATP通过CCNV获取来自ZC的变量信息CCNV\_VariantMessage，该信息可能包含多个line section的变量内容，每个line section变量的结构ST\_VARIANT\_REPORT如Table 4‑15所示。

According to [REF7], ATP software should receive the variants message from the [ZC](#ZC), which may include several line sections' variants. The structure of one line section's variants shows in Table 4‑15.

Table 4‑15 Variant report from [ZC](#ZC)

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | Comments |
| ST\_VARIANT\_REPORT | | | 一个Line变量信息 |
|  | | .LineSectionId | 该变量消息line section编号 |
|  | | .NumberOfVariants | 该line section包括变量个数 |
|  | | .Variants[NumberOfVariants] | 该line section变量 |
|  | | .CreationTime | 变量创建时间 |
|  | .VitalChecksum1 | | SACEM vital checksum 1 |
|  | .VitalChecksum2 | | SACEM vital checksum 2 |

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with LC

### Role of interface

本接口用于ATP通过DVCOM板接收来自LC的轨旁时间、临时限速与版本授权消息，并将ATP的版本报告通过该接口发送给LC。

[iTC\_CC\_ATP-SwRS-0605]

ATP与LC的接口应当遵循文档[REF8]描述，通信配置来自[REF11]。

The interface between ATP and LC shall be compliant with [REF8] document.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0067], [iTC\_CC-SyAD-0068], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0001], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0002], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0005], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0006], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0007], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0008], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0009], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0010], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0067], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0068], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0070], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0011], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0074], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0075], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0076], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0077] , [iTC\_CC\_ATP-Offline-SyID-0012], [iTC\_CC\_ATP-Offline-SyID-0013]

[End]

### Physical level

见文档[REF8]描述。

### Protocol level

见文档[REF8]描述。

### Application level

[iTC\_CC\_ATP-SwRS-0471]

ATP软件每周期查询CCNV是否有转发来自LC的安全消息CCNV\_SynchrodateMessage，CCNV\_VersionAuthMessage和CCNV\_TSRmessage，据此解析生成DateSynchronizationReport，VersionAuthorization和TSRdownloadContent：

* 如果没有LC消息或者CRC校验或SACEM校核字错误，则认为本周期未收到上述消息；
* 否则，分别生成如ST\_DATE\_SYNCH\_REPORT，ST\_VERSION\_AUTH和ST\_TSR\_BLOCK结构的全局变量；
* 其中，如果来自LC的消息中没有[CCNV\_TSRmessage](#CCNV_TSRmessage)，或者[CCNV\_TSRmessage](#CCNV_TSRmessage)结构中的TSR个数超过[REF8]中定义的**MAX\_TSR\_NUMBER**个数，则不生成TSRdownloadContent。

Every cycle, ATP software queries whether there are LC messages [CCNV\_SynchrodateMessage](#CCNV_SynchrodateMessage), [CCNV\_VersionAuthMessage](#CCNV_VersionAuthMessage) and [CCNV\_TSRmessage](#CCNV_TSRmessage) transmitted from CCNV. ATP shall receive and store these messages into [DateSynchronizationReport](#DateSynchronizationReport), [VersionAuthorization](#VersionAuthorization) and [TSRdownloadContent](#TSRdownloadContent):

* If there is no message from LC or the CRC, or vital checksum of the message checked failure, ATP shall discard this message;
* Otherwise, ATP shall generate the structure of global variable such as ST\_DATE\_SYNCH\_REPORT, ST\_VERSION\_AUTH and ST\_TSR\_BLOCK.
* If there is no [CCNV\_TSRmessage](#CCNV_TSRmessage) or the number of TSR in this message is more than **MAX\_TSR\_NUMBER**, ATP shall not generate TSRdownloacContent.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0383], [iTC\_CC-SyAD-0914], [iTC\_CC-SyAD-0961], [iTC\_CC-SyAD-0391], [iTC\_CC-SyAD-0378], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0040], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0041], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0042], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0043], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0044], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0045], [iTC\_CC\_ATP-Offline-SyID-0014]

[End]

#### LC->ATP: Trackside date

ATP根据来自LC的时间信息，维护轨旁设备的时间，并将其发送给ZC使用。

According to the date synchronization information from the LC, the ATP maintains the trackside time, and sends it to [ZC](#ZC).

根据[REF8]，ATP通过CCNV获取来自LC的轨旁时钟信息CCNV\_SynchrodateMessage，其结构ST\_DATE\_SYNCH\_REPORT如Table 4‑16所示。

According to [REF8], ATP software should receive the synchronization message from the LC, as shown in Table 4‑16.

Table 4‑16 Date synchronization report from LC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | | | | Comments |
| ST\_DATE\_SYNCH\_REPORT | | | | 时间同步消息 |
|  | | .Synchrodate | | LC周期时间 |
|  | | .CcLoopHour | | 消息发送时所带CC周期号 |
|  | .VitalChecksum1 | | SACEM vital checksum 1 | |
|  | .VitalChecksum2 | | SACEM vital checksum 2 | |

#### LC->ATP: Temporary speed restriction

在CBTC运用模式下，ATP需考虑线路上设置的临时限速限制。因为当LC刚上电时，需等待来自ATS操作员的TSR解锁命令，在解锁之前，LC发送给线路上所有CC的消息中均无TSR部分。此时，ATP应当认为全线均施加了最严格的临时限速。同样，当ATP长时间收不到来自LC的信息时，也应认为全线设置了TSR。

In CBTC mode, ATP shall consider the temporary speed restriction setting on the track. When LC just power on, the message sent to ATP is not including the TRS parts until the LC received the TRS release command from the ATS operator. At that time, the ATP software shall consider all track map has assigned the most restrictive temporary speed restriction. Similarly, if the ATP has long time not received information from the LC, it shall consider the same restrictive TSR are applied.

根据[REF8]，ATP通过CCNV获取来自LC的临时限速信息CCNV\_TSRmessage，其结构ST\_TSR\_DOWN\_CONTENT如Table 4‑17所示。

According to [REF8], ATP software should receive the TSR message from the LC, as shown in Table 4‑17.

Table 4‑17 TSR download content from LC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | | | | | Comments |
| ST\_TSR\_DOWN\_CONTENT | | | | | 临时限速信息 |
|  | | .NumberOfTSR | | | 临时限速个数 |
|  | | .TSR[numberOfTSR] | | | 每个临时限速内容 |
|  | |  | .Speed | | 临时限速值 |
|  | |  | .FirstBlockId | | 临时限速起始Block |
|  | |  | .StartAbscissa | | 临时限速起始坐标 |
|  | |  | .NumberOfIntermediateBlock | | 临时限速中间的Block个数 |
|  | |  | .IntermediateBlockId[numberOfIntermediateBlock] | | 临时限速中间的Block数组 |
|  | |  | .LastBlockId | | 临时限速结束Block |
|  | |  | .EndAbscissa | | 临时限速结束坐标 |
|  | | .CcLoopHour | | | 消息发送时所带CC周期号 |
|  | .VitalChecksum1 | | | SACEM vital checksum 1 | |
|  | .VitalChecksum2 | | | SACEM vital checksum 2 | |

#### LC->ATP: Version authorization

ATP将自身所使用的软件和数据版本信息通过CCNV发送给LC，LC根据该信息，反馈是否授权CC在相关线路上运行。

The ATP shall send the versions of its software and data to the LC, who uses this information to judge whether to authorize CC running on the relevant track.

根据[REF8]，ATP通过CCNV获取来自LC的版本校验信息CCNV\_VersionAuthMessage，其结构ST\_VERSION\_AUTH如Table 4‑18所示。

According to [REF8], ATP software should receive the version authorization message from the LC, as shown in Table 4‑18.

Table 4‑18 Version authorization from LC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | | | | Comments |
| ST\_VERSION\_AUTH | | | | 版本授权信息 |
|  | | .ZcVitalAuthorization[1...16] | | LC是否授权ATP版本 |
|  | | .ZcNonVitalAuthorization[1...16] | | LC是否授权CCNV版本 |
|  | | .CcLoopHour | | 消息发送时所带CC周期号 |
|  | .VitalChecksum1 | | SACEM vital checksum 1 | |
|  | .VitalChecksum2 | | SACEM vital checksum 2 | |

#### ATP->LC: Version report

根据[REF8]，ATP通过CCNV向LC发送列车版本信息，如所示。

According to [REF8], ATP sends version report to the LC, as shown in .

ATP向LC发送消息的通信由CCNV负责管理。对于ATP，需要将当前所使用的软件版本号、配置数据版本号，存储的线路地图中所有ZC区域的版本号信息，添加ATP的周期时间和校核字后，发送给CCNV。

CCNV is responsible for the communication management with LC. After adding the cycle time and checksum, ATP shall collect and send the information to the CCNV, which include the current software version, the version of the configuration data, the version of the all the [ZC](#ZC) region in the track map,

[iTC\_CC\_ATP-SwRS-0443]

VersionFromCCreport，ATP需要将安全软件、安全配置数据、所有ZC区域的版本号信息以及ATP当前周期号，生成如Table 4‑19格式的消息体，每周期发给CCNV。由CCNV负责与LC的通信。

The ATP shall provide the [VersionFromCCreport](#VersionFromCCreport) message to the CCNV, who will forward them to the LC. The version of the following vital components as :

* ATP software version
* ATP vital parameters version
* ATP used [ZC](#ZC) areas versions (Up to 16 versions maximum)
* ATP cycle time

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0374], [iTC\_CC-SyAD-0375], [iTC\_CC-SyAD-0376], [iTC\_CC-SyAD-0411],

[End]

Table 4‑19 ATP outputs to LC

|  |  |  |
| --- | --- | --- |
| ID | | ATP Outputs |
| ST\_VERSION\_REPORT | | [VersionFromCCreport](#VersionFromCCreport) |
|  | .SafetyParameterVersion | [ATPsetting](#ATPsetting).SafetyParameterVersion |
|  | .SafetyApplicationVersion | [SafeApplicationVersion](#SafeApplicationVersion) |
|  | .SgdVersionZc[1...16] | [TrackMap](#TrackMap).[ZC](#ZC)[1...16].Version |
|  | .CcLoopHour | [ATPtime](#ATPtime)(k) |
|  | .VitalChecksum1 | SACEM vital checksum 1 |
|  | .VitalChecksum2 | SACEM vital checksum 2 |

[iTC\_CC\_ATP-SwRS-0480]

ATP软件每周期将版本报告信息[IdenticalVersionReport](#IdenticalVersionReport)，按照[REF8]定义格式发送给CCNV，由其转发给LC。

Every cycle, ATP software shall send [IdenticalVersionReport](#IdenticalVersionReport) to CCNV according to [REF8]. The CCNV will forward this message to the LC.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0374]

[End]

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with CI Radio

本接口用于ATP通过DVCOM板接收来自CI的增强型后备变量消息，并将计算得到的Overlap可取消信息通过该接口发送给CI。

[iTC\_CC\_ATP-SwRS-0606]

ATP与CI无线通信的接口应当遵循文档[REF5]描述，通信配置来自[REF11]。

The interface between ATP and CI shall be compliant with [REF5] document.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0108], [iTC\_CC-SyAD-1170], [iTC\_CC\_ATP\_SwHA-0234], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0078], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0079], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0082], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0083] , [iTC\_CC\_ATP-Offline-SyID-0012], [iTC\_CC\_ATP-Offline-SyID-0013]

[End]

### Physical level

见文档[REF5]描述。

### Protocol level

ATP和CI无线之间采用SACEM协议进行通信。

Refer to [REF5].

CC和CI间通过SACEM通信协议传输联锁的变量信息和ATP产生的Overlap解锁信息，其过程如Figure 4‑1所示，共有以下四类消息：

* CC→CBI: CBI variant request
* CBI→CC: CBI variant report
* CBI→CC: CC variant request
* CC→CBI: CC variant report

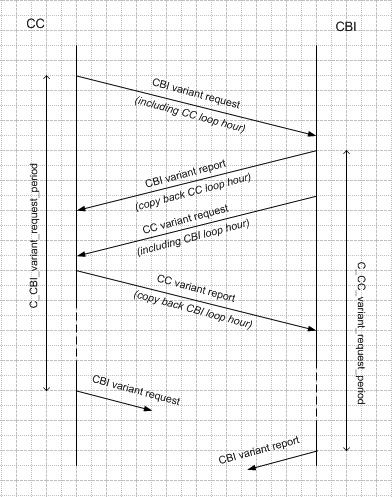


Figure 4‑1 CC-CBI communication with SACEM protocol

### Application level

#### ATP->CI: CBI variant request

对于配置了与联锁无线通信的项目，当选择BM模式后，CCNV会请求向所在区域的联锁发送“CBI variant request”消息，用于请求联锁变量。ATP根据CCNV授权发送的联锁的标识，每周期生成CBI variant request消息并写给CCNV。根据设计，CC最多同时给两个联锁发送变量请求消息。

[iTC\_CC\_ATP-SwRS-0607]

CBIvariantRequest，根据CC-CI接口文档，当来自CCNV的联锁变量ID大于0时，ATP创建CBIvariantRequest消息，本消息安全相关，其内容如Table 4‑20所示。其中校核字的计算参见[REF5]。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0411], [iTC\_CC-SyAD-1170], [iTC\_CC\_ATP\_SwHA-0234], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0082], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0083]

[End]

Table 4‑20 CBI variant request

|  |  |
| --- | --- |
| ID | Description |
| ST\_CBI\_VARIANT\_REQUEST |  |
| .CcLoopHour | [ATPtime](#ATPtime)(k) |
| .VitalChecksum1 | SACEM vital checksum 1 |
| .VitalChecksum2 | SACEM vital checksum 2 |

[iTC\_CC\_ATP-SwRS-0608]

当来自CCNV的联锁变量ID大于0时，向CCNV指定的联锁发送[CBIvariantRequest](#CBIvariantRequest)消息。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1167]

[End]

#### CI->ATP: CBI variant report

CI收到“CBI variant request”消息后，向CC发送“CBI variant report”消息，包括其所辖的变量状态。

[iTC\_CC\_ATP-SwRS-0609]

CBIvariantReport，根据CC-CI接口文档定义，ATP应每周期检查是否有来自CCNV转发的CBI变量报文，其内容如Table 4‑21所示。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1171], [iTC\_CC\_ATP\_SwHA-0234], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0078], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0079]

[End]

Table 4‑21 CBI variant report

|  |  |
| --- | --- |
| ID | Description |
| MessageLength | 消息长度(15...78)，取决于变量个数 |
| NumberOfVariants | 所含变量个数（1...512） |
| ByteVariant[numByteVar] | 以字节表示的CC变量，每个字节最多表示8个比特变量，字节范围（1…64） |
| CcLoopHour | 返回"CBI variant request"中的CC时间 |
| VitalChecksum\_1 | SACEM vital checksum 1 |
| VitalChecksum\_2 | SACEM vital checksum 2 |

#### CI->ATP: CC variant request

CI向CC发送[CBIVariantReport](#CBIVariantReport)消息同时，也向CC发送CCVariantRequest消息

[iTC\_CC\_ATP-SwRS-0610]

CCvariantRequest，根据CC-CI接口文档定义，ATP应每周期检查是否有来自CCNV转发的CC变量请求报文，其内容如Table 4‑22所示。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1193], [iTC\_CC\_ATP\_SwHA-0234], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0080], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0081]

[End]

Table 4‑22 CC variant request

|  |  |
| --- | --- |
| ID | Description |
| CbiLoopHour | 返回"CBI variant request"中的CC时间 |
| VitalChecksum1 | SACEM vital checksum 1 |
| VitalChecksum2 | SACEM vital checksum 2 |

#### ATP->CI: CC variant report

[iTC\_CC\_ATP-SwRS-0611]

CCvariantReport，ATP收到CI的[CCVariantRequest](#CCVariantRequest)后，对于CCNV授权的联锁，回复Overlap解除消息，如Table 4‑23所示。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1195], [iTC\_CC\_ATP\_SwHA-0234], [iTC\_CC\_ATP\_SwHA-0273], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0084], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0085]

[End]

Table 4‑23 CC variant report

|  |  |
| --- | --- |
| ID | Description |
| ST\_CC\_VARIANT\_REPORT |  |
| NumberOfVariants | [TrackMap.NumberOfVariants](#NumberOfVariants)(cbiid) |
| ByteVariant[numByteVar]\* | [CCvariants](#CCvariants)(cbiId, k) |
| CbiLoopHour | [ReceivedCBIloopHour](#ReceivedCBIloopHour)(cbiId, k) |
| VitalChecksum1 | SACEM vital checksum 1 |
| VitalChecksum2 | SACEM vital checksum 2 |

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with PSD

本接口用于ATP通过DVCOM板接收来自CI的PSD状态信息，并将计算得到的PSD控制命令通过该接口发送给CI。

[iTC\_CC\_ATP-SwRS-0612]

ATP与CI的PSD消息接口应当遵循文档[REF5]描述，通信配置来自[REF11]。

The interface between ATP and PSD shall be compliant with [REF5] document.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0067], [iTC\_CC-SyAD-0068], [iTC\_CC-SyAD-0108], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0001], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0002], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0005], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0006], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0007], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0008], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0009], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0010], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0067], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0068], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0070], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0011], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0074], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0075], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0076], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0077], [iTC\_CC\_ATP-Offline-SyID-0015], [iTC\_CC\_ATP-Offline-SyID-0016]

[End]

### Physical level

见文档[REF5]描述。

### Protocol level

CC和CI间通过FSFB2协议传输PSD状态和控制命令，其过程如Figure 4‑2所示，共有以下六种：

* CC→CBI: PSD "Setting"
* CBI→CC: PSD "IO Status"
* CC→CBI: SSE realignment
* CC→CBI: SSR realignment
* CBI→CC: SSE realignment
* CBI→CC: SSR realignment



Figure 4‑2 CC-CBI communication with FSFB2 protocol

### Application level

#### ATP->CI: PSD setting

根据[REF5]，ATP通过CCNV向CI发送屏蔽门控制信息CIsetting，其结构如Table 4‑24所示。

According to [REF5], ATP sends PSD setting message CIsetting to the CI, as shown in Table 4‑24.

ATP发送给CI的PSD控制命令

Table 4‑24 PSD control

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | Logical Type | Description |
| ST\_CI\_SETTING | |  |  |
|  | .PlatformId | NUMERIC\_32 | PSD标识 |
|  | .Order | NUMERIC\_32 | 向联锁发送消息 |

[iTC\_CC\_ATP-SwRS-0478]

根据本周期[CommunicateWithPSD](#CommunicateWithPSD)状态，ATP建立或结束通信的时机如下：

* 如果上周期[CommunicateWithPSD](#CommunicateWithPSD)为**False**，而本周期[CommunicateWithPSD](#CommunicateWithPSD)为**True**，则根据[REF5]定义接口与联锁建立FSFB2通信；
* 否则，如果本周期[CommunicateWithPSD](#CommunicateWithPSD)为**True**，则保持与联锁通信。
* 此时如果与联锁通信中断，则ATP应当重新建立通信。
* 否则，如果本周期[CommunicateWithPSD](#CommunicateWithPSD)为**False**，则结束与联锁的通信。

According to the status of [CommunicateWithPSD](#CommunicateWithPSD), the conditions of establishment are as follow:

* If the [CommunicateWithPSD](#CommunicateWithPSD) at previous cycle was **False** and become **True** in this CYCLE, ATP shall establish connection with CI as the FSFB2 communication protocol.
* Or else:, if the [CommunicateWithPSD](#CommunicateWithPSD) is **True**, ATP shall maintain the connection with CI.
* If the connection with CI is interrupted, ATP shall reestablish the communication.
* Otherwise, if the [CommunicateWithPSD](#CommunicateWithPSD) is **False**, ATP shall end the connection.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0255], [iTC\_CC-SyAD-0949], [iTC\_CC-SyAD-0970], [iTC\_CC-SyAD-1008]

[End]

[iTC\_CC\_ATP-SwRS-0479]

当ATP与联锁已建立通信后，ATP软件每[ATPsetting](#ATPsetting).PSDcommCycle个周期将PSD控制命令[IdenticalCIsetting](#IdenticalCIsetting)根据[REF5]定义的FSFB2接口发送给CCNV，由其转发给联锁：

* 如果[PSDmanagerOrder\_A](#PSDmanagerOrder_A).Id有效，则向联锁发送[IdenticalCIsetting](#IdenticalCIsetting) [0]控制命令；
* 如果[PSDmanagerOrder\_B](#PSDmanagerOrder_B).Id有效，则向联锁发送[IdenticalCIsetting](#IdenticalCIsetting) [1]控制命令。

When the communication between ATP and CI has established, ATP software every [ATPsetting](#ATPsetting).PSDcommCycle cycle sends the PSD control command [IdenticalCIsetting](#IdenticalCIsetting) to CCNV according to [REF5] as FSFB2 protocol.

* IF [PSDmanagerOrder\_A](#PSDmanagerOrder_A).Id is valid, ATP sends command to [IdenticalCIsetting](#IdenticalCIsetting)[0];
* Else: if [PSDmanagerOrder\_B](#PSDmanagerOrder_B).Id is valid, ATP sends command to [IdenticalCIsetting](#IdenticalCIsetting)[1];

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0255], [iTC\_CC-SyAD-0949], [iTC\_CC-SyAD-0970], [iTC\_CC\_ATP\_SwHA-0168], [iTC\_CC-SyAD-1004], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0029], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0030], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0031], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0032]

[End]

#### CI->ATP: PSD IO status

CI发送给ATP的PSD状态如Table 4‑25所示。

Table 4‑25 IO status from CI

|  |  |  |
| --- | --- | --- |
| ID | | Description |
|  | PsdDoorsClosed | Weather the PSD has closed |

[iTC\_CC\_ATP-SwRS-0472]

ATP软件每[ATPsetting](#ATPsetting).PSDcommCycle个周期查询一次CCNV是否有转发来自CI的安全消息，据此解析生成CI\_IOstatus：

* 若当前未收到CI消息，或者CRC校验或FSFB2校核字错误：
* 如果之前与CI通信成功，并收到了有效的消息，且收到该消息未超过**FSFB2\_MESSAGE\_TIMEOUT**个通信周期，则认为该消息仍然有效，据此生成[CI\_IOstatus](#CI_IOstatus)；
* 否则，认为本周期未收到来自CI的消息，设置[CI\_IOstatus](#CI_IOstatus)为限制状态。
* 否则，生成如Table 4‑25的全局变量。

ATP software queries whether there is CI message transmitted from CCNV at every [ATPsetting](#ATPsetting).PSDcommCycle. ATP shall receive and store the message into [CI\_IOstatus](#CI_IOstatus):

* If there is no message from CI, or the CRC, or vital checksum of this message checked failure:
* If the communication with CI succeeded on previous cycles and the age is not more than **FSFB2\_MESSAGE\_TIMEOUT**, ATP shall still use the valid previous CI message to update [CI\_IOstatus](#CI_IOstatus);
* Otherwise, ATP shall set all CI status as restricted.
* Otherwise, ATP shall generate the structure of global variable such as Table 4‑25.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0949], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0048], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0049], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0050], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0051]

[End]

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with DLU

### Role of interface

本接口用于在刚上电后，运行在VLE-2板上的ATP将相关初始化信息通过总线发送给DLU。

This interface is used to ATP sending network addresses to the DLU just after power-up.

[iTC\_CC\_ATP-SwRS-0554]

ATP与DLU的接口应当遵循文档[REF12]描述。

The interface between ATP and DLU shall be compliant with [REF12] document.

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0798], [iTC\_CC\_VLE-2-DLU-SyID-0001], [iTC\_CC\_VLE-2-DLU-SyID-0003]

[End]

### Physical level

Refer to [REF12].

### Protocol level

Refer to [REF12].

### Application level

初始化阶段，ATP需要将Table 4‑26所示内容发送给DLU。

During initialization, ATP software should send the network addressed to DLU as shown in Table 4‑26.

Table 4‑26 ATP output to DLU

|  |  |
| --- | --- |
| ID | Comments |
| DLUIpBlue | DLU's internal IP address in the blue network |
| DLUIpRed | DLU's internal IP address in the red network |

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with Project Data

### Role of interface

本接口用于ATP的离线工具根据工程项目数据生成车载ATP可用的配置参数和线路地图。

This interface is used by the ATP software to read project vital settings and track maps, which generated by CC offline tool from project configurations.

[iTC\_CC\_ATP-SwRS-0555]

ATP与离线配置数据的接口应当遵循文档[REF11]描述。

The interface between ATP and project data shall be compliant with [REF11] document.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1100], [iTC\_CC\_ATP-Offline-SyID-0001], [iTC\_CC\_ATP-Offline-SyID-0002], [iTC\_CC\_ATP-Offline-SyID-0003], [iTC\_CC\_ATP-Offline-SyID-0009] , [iTC\_CC\_ATP-Offline-SyID-0010], [iTC\_CC\_ATP-Offline-SyID-0011]

[End]

### Physical level

见文档[REF11]描述。

Refer to [REF11].

### Protocol level

见文档[REF11]描述。

Refer to [REF11].

### Application level

根据文档[REF11]，ATP的离线工具应当读取并处理项目的数据，生成车载ATP可使用的配置参数及线路地图。项目数据如Table 4‑27所示。

According to the [REF11], the CC offline tool should read project configurations, generate the binary project vital settings and track maps, as shown in Table 4‑27.

Table 4‑27 ATP offline tool inputs from project data

|  |  |  |
| --- | --- | --- |
| ID | Logical type | Comments |
| DATA.VES | Refer to [REF11] | Project data for vital setting, track map and offline generated code |
| CFG.VES | Refer to [REF11] | Project configuration for communication |

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface between two CPUs

### Role of interface

ATP软件运行在VLE-2板上，VLE-2板由2个PowerPC的CPU组成，两块CPU上各运行一套ATP软件，它们所使用的“安全编码处理器”冗余编码不同。两个CPU软件通过VLE-2板上的双口RAM相连，每个周期，它们相互进行时间周期同步。

Two ATP software run on two PowerPC CPUs of the VLE-2 board respectively. They are used different VCP signature tables to avoid common mode failure. The two CPUs connected by dual-ram and software can use this interface to synchronize their cycle time.

### Physical level

见文档[REF9]描述。

Refer to [REF9].

### Protocol level

见文档[REF9]描述。

Refer to [REF9].

### Application level

[iTC\_CC\_ATP-SwRS-0025]

通过VLE-2板上的DPRAM，ATP将Table 4‑28列出的信息写给另一个CPU模块运行的ATP软件。

Through the DPRAM between two CPUS of the VLE-2 board, ATP software should send information listed in Table 4‑28 to the ATP on the other CPU.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0828]

[End]

Table 4‑28 Synchronization data write to the other CPU

|  |  |
| --- | --- |
| ID | Comments |
| [VitalTime](#VitalTime) | 本CPU结束的周期号the current cycle of ATP deferred task |
| [SensorTestFlag](#SensorTestFlag) | CPU1计算的传感器测试标志信息Test flag for sensor testing |
| [VIOM1VitalOut](#VIOM1VitalOut) | 本CPU计算的VIOM1安全输出Vital output to VIOM1 |
| [VIOM2VitalOut](#VIOM2VitalOut) | 本CPU计算的VIOM2安全输出Vital output to VIOM2 |
| [LocReport](#LocReport) | Location report to the [ZC](#ZC) |
| [VersionFromCCreport](#VersionFromCCreport) | Version report to the LC |
| [CCsynchroReport](#CCsynchroReport) | ATP synchronization report to the redundant ATP |
| [CIsetting](#CIsetting) | The PSD setting command to CI |
| [CBIvariantRequest](#CBIvariantRequest) | Request for CBI variants in block mode with radio. |
| [CCvariantReport](#CCvariantReport) | Overlap releasable message to CBI in block mode with radio |

[iTC\_CC\_ATP-SwRS-0026]

通过VLE-2板上的DPRAM，本CPU运行的ATP软件从另一个CPU获取信息如Table 4‑29所示。

Through the DPRAM between two CPUS of the VLE-2 board, ATP software should read information listed in Table 4‑29 from the ATP on the other CPU.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0828]

[End]

Table 4‑29 Synchronization data read from the other CPU

|  |  |
| --- | --- |
| ID | Comments |
| TOC\_VIOM1VitalOut | Vital output to VIOM1 from the other CPU’s ATP |
| TOC\_VIOM2VitalOut | Vital output to VIOm2 from the other CPU’s ATP |
| TOC\_VitalTime | The cycle number from the other CPU’s ATP software |
| TOC\_LocReport | Location report from the other CPU’s ATP software |
| TOC\_VersionReport | Version report from the other CPU’s ATP software |
| TOC\_CCsyncReport | Redundant ATP’s synchronization report from the other CPU’s ATP software |
| TOC\_CIsetting | CI setting from the other CPU's ATP software |
| TOC\_CBIvariantRequest | CBI variant request from the other CPU’s ATP software |
| TOC\_CCvariantReport | CC variant report from the other CPU’s ATP software |

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with Memorized Location

### Role of interface

记忆定位功能用于提高在停车区的效率和安全性，实现列车上电后直接能在ATP保护下运行的功能。列车完全停在休眠区时，ATP可记录该处的位置信息，断电后也不会丢失。当重新上电后，ATP可读取该记录，作为列车的初始化位置进行监控。

Memorized location function used to improve the efficiency and safety in parking are——after the power-on, train can directly running under the ATP protection. When the train completely stopped in sleeping areas, ATP shall record the location and does not lost during power off. And after power-on again, the record can be read as the initialization position of the train.

### Physical level

Will be involved in ATP software design document.

### Protocol level

Will be involved in ATP software design document.

### Application level

如果项目配置了记忆定位（[ATPsetting](#ATPsetting).MemLocAuth），则ATP软件将在入记忆定位作为列车初始化定位。记忆的定位信息MemLocation，如Table 4‑30所示。

The location information need to memorize are shown in Table 4‑30.

Table 4‑30 Memorized location content

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | Logical Type | Description |
|  | .MemLocVersion | NUMERIC\_32 | Version of memorized location implementation |
|  | .SleepAreaId | NUMERIC\_32 | Id of the sleeping zone where train located |
|  | .SleepAreaVersion | NUMERIC\_32 | Version of the sleeping zone where train located |
|  | .TrainType | NUMERIC\_32 | Train type id |
|  | .TrainId | NUMERIC\_32 | Sub system id |
|  | .Ext2 | ST\_LOCATION\_UNIT | External location of End2 |
|  | .Ext1 | ST\_LOCATION\_UNIT | External location of End1 |
|  | .Uncertainty | NUMERIC\_32 | Uncertainty of train location |
|  | .TrainLength | NUMERIC\_32 | Train length |

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

## Interface with VLE Hardware

### Role of interface

ATP软件运行在VLE-2板上，需使用VLE-2板上提供的资源，包括安全时钟、文件系统等。

ATP software runs on the VLE-2 board, so it should use the hardware resources including vital clock, file system etc.

### Physical level

见文档[REF9]描述。

Refer to [REF9].

### Protocol level

见文档[REF9]描述。

Refer to [REF9].

### Application level

[iTC\_CC\_ATP-SwRS-0028]

ATP可获取来自VLE-2板的硬件资源信息如Table 4‑31所示。

ATP can obtain the hardware resources shown in Table 4‑31.

Table 4‑31 ATP inputs from VLE-2

|  |  |
| --- | --- |
| ID | Comments |
| VLE\_GetSafeTime | 来自VLE-2板的安全时钟 |
| VLE\_RTCtime | 来自VLE-2板的非安全时钟(相对于1970年到当前的零时区的秒数) |
| VLE\_RebootRequest | 调用VLE-2板的重启指令 |

#Category= Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0012], [iTC\_CC-SyAD-0067]

[End]

### Software compatibility

N/A

### Hardware compatibility

N/A

### Implicit choices and justification

N/A

# FUNCTIONAL REQUIREMENTS

## Description of Functions

根据车载ATP软件需要完成的功能以其它们之间的层次关系，将ATP软件功能需求划分为如Table 5‑1所示模块。

According to the level of the ATP functionality, the software requirements divided into several functional blocks, as shown in Table 5‑1.

Table 5‑1 Functions table

|  |  |  |
| --- | --- | --- |
| Functional blocks | | Description |
| F1- Manage System Information | | 管理列车状态 |
|  | F11-Acquire Configuration Data | 获取配置数据 |
|  | F12-Manage Train Status | 采集列车输入 |
|  | F13-Manage Loop Hour with Distant ATP | 与远端ATP通信 |
|  | F14-Manage Variants in Block Mode | 后备模式下变量管理 |
|  | F15-Manage Variants in CBTC Mode | CBTC模式下变量管理 |
| F2-Measure Train Kinematics | | 列车运动学参数计算 |
|  | F21-Manage Interface with | 与VPB板接口 |
|  | F22-Monitor the Odometer | 监测编码里程计 |
|  | F23-Manage the Odometer State | 管理编码里程计状态 |
|  | F24-Compensate Sliding Slipping Effect | 打滑补偿 |
|  | F25-Calculate Radar Speed | 计算来自雷达的速度信息 |
|  | F26-Detect Odometer Axle Lock | 轴锁检测 |
|  | F27-Compute Train Kinematics | 计算列车运动学参数 |
|  | F28-Calibrate Wheel Movement | 轮径校准 |
| F3-Locate the Train on Track Map | | 在线路中定位 |
|  | F31-Initialize Train Location | 初始化列车定位 |
|  | F32-Update Train Location | 更新列车定位 |
|  | F33-Confirm Train Localization | 管理列车定位状态 |
| F4-Monitor Train Energy | | 监控列车能量 |
|  | F41-Determine the EOA | 计算列车授权运行位置 |
|  | F42-Manage Coerced Permissive or Restrictive | 强制允许或强制限制监控 |
|  | F43-Manage Temporary Speed Restriction | 管理临时限速 |
|  | F44-Compute Train Energy | 计算列车能量 |
|  | F45-Process Singularities | 限制奇点处理 |
|  | F46-Determine Over Energy | 判断列车是否超能 |
| F5-Monitor Train Position and Speed | | 监控列车位置和速度 |
|  | F51-Moral Time | 信号机模糊时间监控 |
|  | F52-Prevent Train Moving in Undetectable Danger | 不可侦测风险监控 |
|  | F53-Monitor Train Speed | 监控列车速度 |
|  | F54-Monitor Rollback Train Speed | 监控后溜 |
|  | F55-Monitor Reverse Train Speed | 监控倒车 |
| F6-Protect Passengers Entrance and Exit from the Train | | 车门管理相关 |
|  | F61-Elaborate Door Opening Authorization | 确认门授权 |
|  | F62-Control PSD Opening and Closing Order | 控制屏蔽门命令 |
|  | F63-Monitor Status of Doors | 监控门状态 |
|  | F64-Protect Passengers during Emergency Evacuation | 紧急逃生管理 |
| F7-Generate Output Orders | | 生成对外部系统的输出 |
|  | F71-Outputs to Rolling-stock | 计算列车输出 |
|  | F72-Outputs to ZC | 计算ZC的位置报告 |
|  | F73-Outputs to CI Radio | 计算联锁无线通信报文 |
| F8-Platform Relative & Assist Functions | | 平台相关辅助功能 |
|  | F81-Initialize ATP software | 软件初始化 |
|  | F82-Manage Vital Time | 管理安全时钟 |
|  | F83-Manage Synchronization between two CPUs | 上下CPU软件同步 |

## Description of Data

### Constants for iTC production

车载ATP作为iTC系统的一部分，应遵从iTC系统的容量限制。因此，ATP软件能够处理Table5-2中所中所定义规模的外部系统信息。

As a part of iTC, ATP shall comply with the capacity limit of the system. Therefore, ATP software shall able to handle such a scale of information defined in Table5-2.

Table 5‑2 Constants for iTC production

|  |  |  |
| --- | --- | --- |
| Identification | Value | Description |
| **MAX\_BEACON\_NB** | 7000 | Maximum number of beacons supported by the iTC production |
| **MAX\_BLOCK\_NB** | 3000 | Maximum number of blocks supported by the iTC production |
| **MAX\_CBI\_NB** | 32 | Maximum number of interlocks supported by the iTC production |
| **MAX\_LC\_NB** | 16 | Maximum number of line controller supported by the iTC production |
| **MAX\_LINE\_SECTION\_NB** | 20 | Maximum number of line section supported by the iTC production |
| **MAX\_PLATFORM\_NB** | 500 | Maximum number or platforms supported by the iTC production |
| **MAX\_ZC\_NB** | 16 | Maximum number or zone controller supported by the iTC production |

### Configurable data for projects

为支持不同项目的情况，ATP软件在运行前会载入可由项目配置的安全设置参数，包含了线路属性、项目要求、列车运动学特性等内容，其结构如Table 6‑1所示。其详细定义见[REF11]。

For application to different projects, ATP shall load vital settings before cycling operation, as shown as Table 6‑1, the configurable data including the parameters of track, specific demands for the project, train characteristics etc. Refer to [REF11] for the detail definition.

### Track map

车载ATP运行时使用的线路地图来自项目配置。如Figure 5‑1所示，线路地图以block为单位组织，由ZC分组管理；在block上，有一系列[奇点](#ENUM_SINGULARITY_TYPE)，分别表示信号机、道岔、站台等实际的物理设备，以及永久限速、保护区等虚拟限制区域或限制点。

The track map onboard ATP used come from project configuration. As shown in Figure 5‑1, the basic constitution unit of track map is block, which managed by ZC; on the block, a series of [singularities](#ENUM_SINGULARITY_TYPE) represent physical devices such as signal, switcher, platform, as well as permanent speed restrictions, protected areas and other virtual restricted zones or limit points.



Figure 5‑1 Track map layout for project

按照线路地图的设计，每个BLOCK上的坐标是相对于该BLOCK上行方向起始点的距离值，如Figure 5‑2所示。。上行方向的BLOCK起点坐标为0，并依次递增直到BLOCK长度。BLOCK下行方向的起点坐标是该BLOCK的最大值，即该BLOCK的长度。如果向下行方向运行，坐标逐渐减小，直到0为止。就是说，在同一个BLOCK上，越往上行坐标越大，反之亦然。如果一个坐标值超过某BLOCK长度，则实际位置应当在该BLOCK上行方向的下游BLOCK上；反之如果坐标小于0，则实际位置应当在该BLOCK下行方向的下游BLOCK上。

According to the design of track map, an abscissa of a block means the distance from the block endpoint of the **UP** orientation, as shown in Figure 5‑2. The abscissa of the block starts from zero, the **UP** orientation endpoint, and increases along the **UP** orientation until reached the length of this block, the **DOWN** orientation endpoint. That is, in a same block, towards **UP** orientation, the larger of the abscissa value, the upper of the location; and vice versa. If an abscissa exceeds the length of the block, the actual location should be in the downstream block on the **UP** orientation; other hand, if an abscissa is less than zero, it should be in the downstream block on the **DOWN** orientation.



Figure 5‑2 Abscissa increasing rule in block

线路的上行和下行方向，由项目而定。

The **UP** or **DOWN** orientation in track map is defined by project.

对于线路上的部分奇点，其状态是会发生动态变化的，例如道岔的位置，信号机为允许或限制等。对于此类奇点，在线路地图中会标有指定的变量作为其状态变化的索引。而这些变量，会通过轨旁设备发送给车载ATP：在CBTC模式下，ATP使用来自ZC发送的变量；而在Block模式下，ATP使用来自BM信标发送的变量，或者使用来自CBI发送的无线变量（如果该项目有无线通信的Block模式）。变量的索引定义如Table 5‑3所示。

Some of the singularities, there status will change dynamically, such as switch positions, permissive or restrictive signals.

Table 5‑3 Index of variant

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identification | | | Logical Type | Description |
| Variant | | | | |
|  | LineSec | | | Variant index in line section of ZC |
|  |  | Id | NUMERIC\_32 | The line section id |
|  |  | Index | NUMERIC\_32 | The index in line section |
|  | Cbi | | | Variant index in CBI for block mode with radio |
|  |  | Id | NUMERIC\_32 | The CBI id |
|  |  | Index | NUMERIC\_32 | The index in Cbi |

## F1- Manage System Information

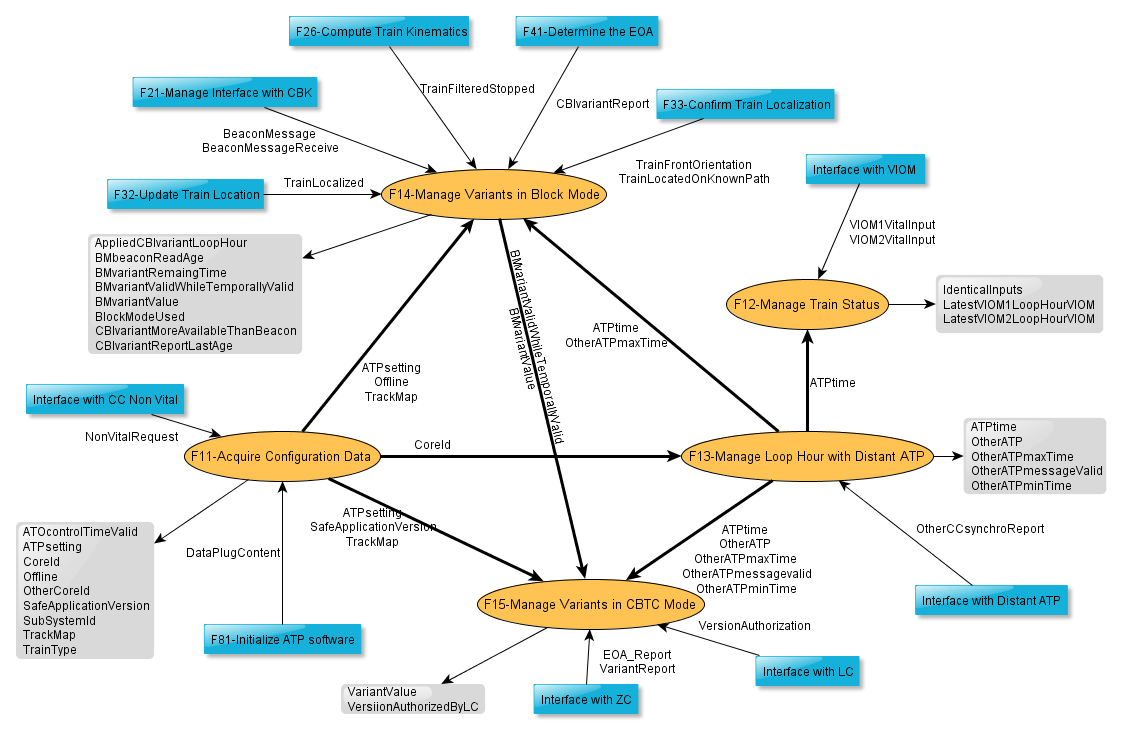


Figure 5‑3 SART modeling of function F1

## F11-Acquire Configuration Data

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [DataPlugContent](#DataPlugContent) | Internal | F81-Initialize ATP software |
| [NonVitalRequest](#NonVitalRequest) | External | Interface with CC Non Vital |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [IdenticalVersionOfDualCPU](#IdenticalVersionOfDualCPU) | √ | √ |  |
| [NonVitalRequestReady](#NonVitalRequestReady) | × | √ | BOOLEAN |
| [TrainKnown](#TrainKnown) | √ | √ | BOOLEAN |

### Processing

#### Get CC data plug information

同一项目中，运行在不同车辆的CC上的ATP软件本身是相同的。在与外部系统通信时，依靠CC SSID来区分当前的CC标识；同时，ATP还需知道自身所在车头是**END\_1**还是**END\_2**；以及运行在VLE-2板的哪个CPU模块。上述信息均需通过读取安装在VLE-2板上的CC data plug获取。须由操作人员保证，安装在每一块VLE-2板上的CC data plug都是正确的，唯一的。

In the same project, the ATP software running on different vehicles is identical. When communicating with external systems, the ATP depends on the CC SSID to identify itself; meanwhile, it needs to know itself in the train **END\_1** or **END\_2**; and runs in the CPU1 or CPU2 module of the VLE-2 board. These information are stored in CC data plug where installed on each CPU module of the VLE-2 board. The maintenance staff guarantees the correctness and uniqueness of CC data plug.

[iTC\_CC\_ATP-SwRS-0053]

初始化时，ATP读取来自CC data plug的[DataPlugContent](#DataPlugContent).CCTrainType信息，生成TrainType。

On initialization, ATP generates [TrainType](#TrainType) according to [DataPlugContent](#DataPlugContent).CCTrainType from the CC data plug.

def [TrainType](#TrainType)(k):

return [DataPlugContent](#DataPlugContent).CCTrainType

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0371]

[End]

[iTC\_CC\_ATP-SwRS-0054]

初始化时，ATP读取来自CC data plug的[DataPlugContent](#DataPlugContent).CCCoreId信息，生成CoreId。

On initialization, ATP generates [CoreId](#CoreId) according to [DataPlugContent](#DataPlugContent).[CCCoreId](#CCCoreId) read from the CC data plug.

def [CoreId](#CoreId)(k):

return [DataPlugContent](#DataPlugContent).CCCoreId

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0371]

[End]

[iTC\_CC\_ATP-SwRS-0613]

OtherCoreId，远端车头号

Core id for CC on the distant cab.

def [OtherCoreId](#OtherCoreId)(k):

if ([CoreId](#CoreId)(k) is **END\_1**):

return **END\_2**

elif ([CoreId](#CoreId)(k) is **END\_2**):

return **END\_1**

else:

return **None**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0371]

[End]

[iTC\_CC\_ATP-SwRS-0055]

初始化时，ATP读取来自CC data plug的[DataPlugContent](#DataPlugContent).CC\_SSID信息，生成SubSystemId。

On initialization, ATP generates [SubSystemId](#SubSystemId) according to [DataPlugContent](#DataPlugContent).CC\_SSID from the CC data plug.

def [SubSystemId](#SubSystemId)(k):

return [DataPlugContent](#DataPlugContent).CC\_SSID

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0210], [iTC\_CC-SyAD-0371]

[End]

#### Check the project setting

ATP软件适用于不同的工程项目，对于项目相关的配置参数和线路地图等信息，须由离线工具生成，并存储在VLE-2板指定存储空间。ATP软件在上电后通过读取配置参数和线路地图来获取当前项目信息，根据配置项执行相关功能。ATP软件应当通过校验来保证配置参数和线路地图是正确的。对于相同类型的列车，冗余ATP和不同CPU模块之间的ATP所使用的配置参数和线路地图也应当是相同的。

ATP software should apply to different projects. For project-related configuration parameters, track maps and other information, generated by CC offline tool, are stored in the VLE-2 board designated storage space. When power on, the ATP software shall read these configurations and perform for related functions. Before using, the ATP shall verify the correctness of these parameters and track maps, and shall ensure the consistency of these data with the other CPU module and with the redundant ATP on the other END of train.

[iTC\_CC\_ATP-SwRS-0057]

根据[TrainType](#TrainType)类型，车载ATP读取相对应由离线工具生成的带VCP编码的ATP项目配置参数[DATA.VES](#DATAVES)，并进行校验，如果校验正确则生成车载ATP配置数据ATPsetting，其结构如Table 6‑1所示；若校验错误则ATP将不继续运行。

According with train type, ATP software shall compute the VCP check words of the project vital setting [DATA.VES](#DATAVES), which generated by CC offline tools.

* If check words are correct, ATP software shall read the vital setting and convert to the global variable [ATPsetting](#ATPsetting) according to the Table 6‑1;
* Otherwise, the ATP shall cease to run.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0332], [iTC\_CC-SyAD-0385], [iTC\_CC\_ATP\_SwHA-0011], [iTC\_CC\_ATP-Offline-SyID-0001], [iTC\_CC\_ATP-Offline-SyID-0002], [iTC\_CC\_ATP-Offline-SyID-0003], [iTC\_CC\_ATP-Offline-SyID-0004], [iTC\_CC\_ATP-Offline-SyID-0009], [iTC\_CC-SyAD-0032]

[End]

[iTC\_CC\_ATP-SwRS-0058]

根据[TrainType](#TrainType)类型，车载ATP读取由离线工具生成的带VCP编码的线路地图数据[DATA.VES](#DATAVES)，并进行校验，如果校验正确则生成TrackMap，其结构见[REF11]定义；如果校验错误，则ATP将不继续运行。

According with train type, ATP software shall read [DATA.VES](#DATAVES) and detect the track map data with VCP which was generated by CC offline tools. If the data is correct, ATP shall generate [TrackMap](#TrackMap) with the structure as refer to [REF11]; Otherwise, ATP cannot continue to operate.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0332], [iTC\_CC-SyAD-0385], [iTC\_CC\_ATP\_SwHA-0174], [iTC\_CC\_ATP-Offline-SyID-0001], [iTC\_CC\_ATP-Offline-SyID-0002], [iTC\_CC\_ATP-Offline-SyID-0003], [iTC\_CC\_ATP-Offline-SyID-0005], [iTC\_CC\_ATP-Offline-SyID-0009], [iTC\_CC-SyAD-0032]

[End]

[iTC\_CC\_ATP-SwRS-0755]

[DATA.VES](#DATAVES)中部分与输入输出相关功能的代码，由离线工具根据项目安全配置数据生成，带有VCP编码，ATP在初始化时读取使用，生成Offline结构如Table 6‑3所示，详见[REF11]。

Some parts of codes in [DATA.VES](#DATAVES), related to input and output functions, as project configuration data, ATP links them during initialization and generates Offline structure as Table 6‑3, refer to [REF11].

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0332], [iTC\_CC-SyAD-0385], [iTC\_CC-SyAD-0032], [iTC\_CC\_ATP\_SwHA-0174], [iTC\_CC\_ATP\_SwHA-0189], [iTC\_CC\_ATP\_SwHA-0190], [iTC\_CC\_ATP\_SwHA-0200], [iTC\_CC\_ATP\_SwHA-0201], [iTC\_CC\_ATP\_SwHA-0202], [iTC\_CC\_ATP\_SwHA-0203], [iTC\_CC\_ATP\_SwHA-0205], [iTC\_CC\_ATP\_SwHA-0208], [iTC\_CC\_ATP\_SwHA-0233], [iTC\_CC\_ATP\_SwHA-0253], [iTC\_CC\_ATP\_SwHA-0265], [iTC\_CC\_ATP\_SwHA-0268], [iTC\_CC\_ATP\_SwHA-0269], [iTC\_CC\_ATP-Offline-SyID-0001], [iTC\_CC\_ATP-Offline-SyID-0002], [iTC\_CC\_ATP-Offline-SyID-0003], [iTC\_CC\_ATP-Offline-SyID-0006], [iTC\_CC\_ATP-Offline-SyID-0007], [iTC\_CC\_ATP-Offline-SyID-0008], [iTC\_CC\_ATP-Offline-SyID-0009]

[End]

**NOTE：**

对于离线工具生成的项目配置参数和线路地图，如果在车载ATP软件读取发生错误，或者在运行过程中该配置的值发生了改变，则会使得VCP编码计算错误，从而导致VIOM输出为限制状态，导向安全。

For the configuration parameters and track map, generated by the CC offline tool, if there is error for the ATP reading, or there is any changes during the processing, the VCP coding calculation will be conducted the wrong data, which will causing the VIOM output is set as restricted status.

[iTC\_CC\_ATP-SwRS-0536]

SafeApplicationVersion，ATP软件的版本号。ATP软件应当在代码中定义软件的版本号，每次修改代码时，更新该版本号。

[SafeApplicationVersion](#SafeApplicationVersion) stands for the version of ATP software. The version of ATP software is defined in source code. The version shall be update whenever the code changed.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0388]

[End]

[iTC\_CC\_ATP-SwRS-0537]

IdenticalVersionOfDualCPU，ATP软件在获取了data plug和离线数参数及线路地图后，需与另一个模块进行比较，如果下列内容全都一致，则设置IdenticalVersionOfDualCPU为**True**；否则如果任意一项不相同，则设置IdenticalVersionOfDualCPU为**False**。

* [TrainType](#TrainType)，来自data plug的列车类型；
* [CoreId](#CoreId)，来自data plug的所在车头标识；
* [SubSystemId](#SubSystemId)，来自data plug的列车标识；
* [TrackMap](#TrackMap)，线路地图中相应每个ZC区的版本号；
* [ATPsetting](#ATPsetting)，配置数据的版本号；
* [SafeApplicationVersion](#SafeApplicationVersion)，ATP软件版本号。

After ATP gets the offline parameters and track map from the CC data plug, it needs to compare this information with the other CPU module, which represented by the term of [IdenticalVersionOfDualCPU](#IdenticalVersionOfDualCPU). If the information is as same, ATP shall set [IdenticalVersionOfDualCPU](#IdenticalVersionOfDualCPU) as **True**; otherwise, it will set [IdenticalVersionOfDualCPU](#IdenticalVersionOfDualCPU) as **False.**

* [TrainType](#TrainType), the train type from data plug;
* [CoreId](#CoreId), the train end identification from data plug;
* [SubSystemId](#SubSystemId), the train identification from data plug;
* [TrackMap](#TrackMap), the version number of each [ZC](#ZC) region in line map;
* [ATPsetting](#ATPsetting), the version number for the configuration data;
* [SafeApplicationVersion](#SafeApplicationVersion), the version of ATP software.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0388], [iTC\_CC\_ATP\_SwHA-0173], [iTC\_CC-SyAD-0032]

[End]

[iTC\_CC\_ATP-SwRS-0059]

上述信息均获取正确并且相一致后，生成TrainKnown信息。如果TrainKnown为**False**，则VIOM将输出全限制状态。

After all above-mentioned information has corrected, and correspondingly, ATP will generate [TrainKnown](#TrainKnown) information. If [TrainKnown](#TrainKnown) considered as **False**, ATP shall set all output ports as restricted.

def [TrainKnown](#TrainKnown)(k):

return (([TrainType](#TrainType)(k) == [ATPsetting](#ATPsetting).TrainTypeId)

and ([CoreId](#CoreId)(k) is **END\_1**

or [CoreId](#CoreId)(k) is **END\_2**)

and [IdenticalVersionOfDualCPU](#IdenticalVersionOfDualCPU)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0332], [iTC\_CC-SyAD-0333], [iTC\_CC-SyAD-0372], [iTC\_CC\_ATP\_SwHA-0006], [iTC\_CC\_ATP\_SwHA-0007], [iTC\_CC-SyAD-0032]

[End]

#### Parse the CCNV information

[iTC\_CC\_ATP-SwRS-0125]

NonVitalRequestReady，通过与CCNV的通信接口，判断是否收到CCNV的消息[NonVitalRequest](#NonVitalRequest)

Through the communication with CCNV, ATP judges [NonVitalRequest](#NonVitalRequest)received from CCNV and generates [NonVitalRequestReady](#NonVitalRequestReady) If received a new message.

def NonVitalRequestReady(k):

return Message.[Exists](#Exists)([NonVitalRequest](#NonVitalRequest))

#Category= Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0068]

[End]

[iTC\_CC\_ATP-SwRS-0126]

ATOcontrolTimeValid，CCNV消息有效标志，如果超过**CCNV\_VALIDITY\_CYCLES**个周期仍未收到新的CCNV消息，则设置为**False**。

[ATOcontrolTimeValid](#ATOcontrolTimeValid) stands for the effectiveness of CCNV message. If there is no updating CCNV message past the **CCNV\_VALIDITY\_CYCLES**, [ATOcontrolTimeValid](#ATOcontrolTimeValid) is set as **False**.

def [ATOcontrolTimeValid](#ATOcontrolTimeValid)(k):

if ([NonVitalRequestReady](#NonVitalRequestReady)(k)):

[ATOcontrolTimeValid](#ATOcontrolTimeValid) = **True**

[ATOcontrolTimer](#ATOcontrolTimer) = 0

elif ([ATOcontrolTimer](#ATOcontrolTimer)(k-1) < CCNV\_VALIDITY\_CYCLES):

[ATOcontrolTimer](#ATOcontrolTimer) = [ATOcontrolTimer](#ATOcontrolTimer)(k-1) + 1

else:

[ATOcontrolTimeValid](#ATOcontrolTimeValid) = **False**

return [ATOcontrolTimeValid](#ATOcontrolTimeValid)

#Category= Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1044]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [ATOcontrolTimeValid](#ATOcontrolTimeValid) | × | √ | BOOLEAN |
| [ATPsetting](#ATPsetting) | √ | × | [REF11] |
| [CoreId](#CoreId) | √ | √ | NUMERIC\_32 |
| [Offline](#Offline) | √ | × | [REF11] |
| [OtherCoreId](#OtherCoreId) | √ | √ | NUMERIC\_32 |
| [SafeApplicationVersion](#SafeApplicationVersion) | √ | √ | NUMERIC\_32 |
| [SubSystemId](#SubSystemId) | √ | √ | NUMERIC\_32 |
| [TrackMap](#TrackMap) | √ | × | [REF11] |
| [TrainType](#TrainType) | √ | √ | NUMERIC\_32 |

## F12-Manage Train Status

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPtime](#ATPtime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [VIOM1VitalInput](#VIOM1VitalInput) | External | Interface with VIOM |
| [VIOM2VitalInput](#VIOM2VitalInput) | External | Interface with VIOM |

### Locals

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Safety Critical | | Observable | | Logical Type | |
| [VIOM1VitalInputsAvailable](#VIOM1VitalInputsAvailable) | | √ | | √ | | BOOLEAN |
| [VIOM1VitalInputsLastAge](#VIOM1VitalInputsLastAge) | | √ | | √ | | NUMERIC\_32 |
| [VIOM1VitalInputsReceived](#VIOM1VitalInputsReceived)[2] | | √ | | √ | | BOOLEAN |
| [VIOM1VitalInputsValid](#VIOM1VitalInputsValid) | | √ | | √ | | BOOLEAN |
| [VIOM2VitalInputsAvailable](#VIOM2VitalInputsAvailable) | | √ | | √ | | BOOLEAN |
| [VIOM2VitalInputsLastAge](#VIOM2VitalInputsLastAge) | | √ | | √ | | NUMERIC\_32 |
| [VIOM2VitalInputsReceived](#VIOM2VitalInputsReceived)[2] | | √ | | √ | | BOOLEAN |
| [VIOM2VitalInputsValid](#VIOM2VitalInputsValid) | | √ | | √ | | BOOLEAN |

### Processing

ATP软件每周期查询CCNV是否有转发来自VIOM的安全消息，包括来自**END\_1**和**END\_2**端车头的各2路VIOM。消息结构如Table 4‑11，ATP软件将其解析生成VIOM1VitalInput，或VIOM2VitalInput，规则如下：

* 如果没有收到某个VIOM的消息，或者CRC校验或SACEM校核字错误，则将丢弃该VIOM信息；
* 否则，生成如ST\_VIOM\_INPUT结构的全局变量。

Every cycle, ATP software queries whether there are vital rolling stock inputs messages transmitted from CCNV, which acquired by at most four VIOM from both **END\_1** and **END\_2**. The structure of the inputs message are Table 4‑11, and ATP shall receive and store them into VIOM1VitallInput.or VIOM2VitalInput. The receiving and storing rules are following:

* If there is no message from VIOM, or the CRC or vital checksum of the received message failed, ATP shall discard this message;
* Otherwise, generate the structure of global variable such as ST\_VIOM\_INPUT.

#### VIOM1 -> ATP

[iTC\_CC\_ATP-SwRS-0468]

VIOM1VitalInputsReceived，ATP判断是否收到了来自VIOM1的安全输入消息。

ATP determines whether received a safety input message from VIOM1.

def [VIOM1VitalInputsReceived](#VIOM1VitalInputsReceived)(viomId, k):

return [Message.Received](#Received)([VIOM1VitalInput](#VIOM1VitalInput)(viomId), k)

其中viomId取值为0或1，表示位于**END\_1**车头2个VIOM中的1个。

During the calculation, the value viomId is either zero or one, which represents one of the two VIOM in the train **END\_1**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0031], [iTC\_CC-SyAD-0961], [iTC\_CC\_ATP\_SwHA-0017], [iTC\_CC-SyAD-0988], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0036], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0037]

[End]

[iTC\_CC\_ATP-SwRS-0060]

VIOM1VitalInputsAvailable，通过通信接口，获取来自VIOM1的安全输入消息，并判断消息传输的时间有效性以及顺序的正确性。

Through the communication, ATP gets the vital input message from VIOM1 and decides the time effectiveness and the correctness of the sequence of the message, which defined as [VIOM1VitalInputsAvailable](#VIOM1VitalInputsAvailable).

def [VIOM1VitalInputsAvailable](#VIOM1VitalInputsAvailable)(ViomId, k):

return Message.[Available](#Available)([VIOM1VitalInputsReceived](#VIOM1VitalInputsReceived)(ViomId, k),

[VIOM1VitalInput](#VIOM1VitalInput)(ViomId).AtpLoopHour,

**VIOM\_VALIDITY\_TIME**,

[VIOM1VitalInputsLastAge](#VIOM1VitalInputsLastAge)(ViomId, k-1),

k)

其中ViomId取值为0或1，表示位于**END\_1**车头2个VIOM中的1个。

During the calculation, the value ViomId is either zero or one, which represents one of the two VIOM in the train **END\_1**.

#Category= Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0988], [iTC\_CC\_ATP\_SwHA-0017]

[End]

[iTC\_CC\_ATP-SwRS-0740]

VIOM1VitalInputsLastAge，记录收到最新的**END\_1**端VIOM的存活时间为多少。

Records the survival time of received vital inputs from VIOM1.

def [VIOM1VitalInputsLastAge](#VIOM1VitalInputsLastAge)(ViomId, k):

return Message.[LastAge](#LastAge)([VIOM1VitalInputsAvailable](#VIOM1VitalInputsAvailable)(ViomId, k),

[VIOM1VitalInput](#VIOM1VitalInput)(ViomId).AtpLoopHour,

[VIOM1VitalInputsLastAge](#VIOM1VitalInputsLastAge)(ViomId, k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0987], [iTC\_CC-SyAD-0988], [iTC\_CC-SyAD-1005]

[End]

[iTC\_CC\_ATP-SwRS-0572]

LatestVIOM1LoopHourVIOM，记录当前收到最新的**END\_1**端VIOM的周期时间信息。

* 初始化时LatestVIOM1LoopHourVIOM为VIOM周期号的最小值0；
* 如果收到可用的VIOM1信息，或之前的VIOM1消息已无效但又新收到一条VIOM1消息，则将相应的LatestVIOM1LoopHourVIOM设置为新收到消息中的viomLoopHour值；
* 否则，LatestVIOM1LoopHourVIOM保持不变。

ATP records the latest cycle time information of VIOM in **END\_1** by the term [LatestVIOM1LoopHourVIOM](#LatestVIOM1LoopHourVIOM).

* In initialization, set [LatestVIOM1LoopHourVIOM](#LatestVIOM1LoopHourVIOM) as the zero;
* If receiving an available VIOM1 message, or a new message and the previous one has invalid, ATP will set the related value of [LatestVIOM1LoopHourVIOM](#LatestVIOM1LoopHourVIOM) as the viomLoopHour of the message.
* Otherwise, [LatestVIOM1LoopHourVIOM](#LatestVIOM1LoopHourVIOM) keeps unchanged.

def [LatestVIOM1LoopHourVIOM](#LatestVIOM1LoopHourVIOM)(ViomId, k):

if ([VIOM1VitalInputsAvailable](#VIOM1VitalInputsAvailable)(k)

or (not [VIOM1VitalInputsValid](#VIOM1VitalInputsValid)(k-1)

and [VIOM1VitalInputsReceived](#VIOM1VitalInputsReceived)(ViomId, k))):

return [VIOM1VitalInput](#VIOM1VitalInput)[ViomId].ViomLoopHour

else:

return [LatestVIOM1LoopHourVIOM](#LatestVIOM1LoopHourVIOM)[ViomId](k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0987], [iTC\_CC-SyAD-0988], [iTC\_CC-SyAD-1005]

[End]

#### VIOM2 -> ATP

[iTC\_CC\_ATP-SwRS-0614]

VIOM2VitalInputsReceived，收到并校验正确来自VIOM2的安全输出消息。

ATP determines whether received a safety input message from VIOM2.

def [VIOM2VitalInputsReceived](#VIOM2VitalInputsReceived)(ViomId, k):

return [Message.Received](#Received)([VIOM2VitalInput](#VIOM2VitalInput)(ViomId), k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0031], [iTC\_CC-SyAD-0961], [iTC\_CC\_ATP\_SwHA-0017], [iTC\_CC-SyAD-0988], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0036], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0037]

[End]

[iTC\_CC\_ATP-SwRS-0449]

VIOM2VitalInputsAvailable，通过通信接口，获取来自VIOM2的安全输入消息，并判断消息传输的时间有效性以及顺序的正确性。

Through the communication, ATP gets the vital input message from VIOM2 and decides the time effectiveness and the correctness of the sequence of the message, which defined as [VIOM2VitalInputsAvailable](#VIOM2VitalInputsAvailable).

def [VIOM2VitalInputsAvailable](#VIOM2VitalInputsAvailable)(ViomId, k):

return Message.[Available](#Available)([VIOM2VitalInputsReceived](#VIOM2VitalInputsReceived)(ViomId, k),

[VIOM2VitalInput](#VIOM2VitalInput)(ViomId).AtpLoopHour,

**VIOM\_VALIDITY\_TIME**,

[VIOM2VitalInputsLastAge](#VIOM2VitalInputsLastAge)(ViomId, k-1),

k)

其中ViomId取值为0或1，表示位于**END\_2**车头2个VIOM中的1个。

During the calculation, the value i is either zero or one, which represents one of the two VIOM in the train **END\_2**.

#Category= Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0988], [iTC\_CC\_ATP\_SwHA-0017], [iTC\_CC-SyAD-1005]

[End]

[iTC\_CC\_ATP-SwRS-0741]

VIOM2VitalInputsLastAge，记录当前收到最新的**END\_2**端VIOM的周期时间信息。

Records the survival time of received vital inputs from VIOM2.

def [VIOM2VitalInputsLastAge](#VIOM2VitalInputsLastAge)(viomId, k):

return Message.[LastAge](#LastAge)([VIOM2VitalInputsAvailable](#VIOM2VitalInputsAvailable)(viomId, k),

[VIOM2VitalInput](#VIOM2VitalInput)(viomId).AtpLoopHour,

[VIOM2VitalInputsLastAge](#VIOM2VitalInputsLastAge)(viomId, k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0987], [iTC\_CC-SyAD-0988]

[End]

[iTC\_CC\_ATP-SwRS-0573]

LatestVIOM2LoopHourVIOM，记录当前收到最新的**END\_2**端VIOM的周期时间信息。

* 初始化时LatestVIOM2LoopHourVIOM为VIOM周期号的最小值0；
* 如果收到可用的VIOM2信息，或之前的VIOM2消息已无效但又新收到一条VIOM2消息，则将相应的LatestVIOM2LoopHourVIOM设置为新收到消息中的ViomLoopHour值；
* 否则，LatestVIOM2LoopHourVIOM保持不变。

ATP records the latest cycle time information of VIOM in **END\_2**by the term [LatestVIOM2LoopHourVIOM](#LatestVIOM2LoopHourVIOM).

* In initialization, set [LatestVIOM2LoopHourVIOM](#LatestVIOM2LoopHourVIOM) as the zero;
* If receiving an available VIOM1 message, or a new message and the previous one has invalid, ATP will set the related value of [LatestVIOM2LoopHourVIOM](#LatestVIOM2LoopHourVIOM) as the viomLoopHour of the message.
* Otherwise, [LatestVIOM2LoopHourVIOM](#LatestVIOM2LoopHourVIOM) keeps unchanged.

def [LatestVIOM2LoopHourVIOM](#LatestVIOM2LoopHourVIOM)(ViomId, k):

if ([VIOM2VitalInputsAvailable](#VIOM2VitalInputsAvailable)(k)

or (not [VIOM2VitalInputsValid](#VIOM2VitalInputsValid)(k-1)

and [VIOM2VitalInputsReceived](#VIOM2VitalInputsReceived)(ViomId, k))):

return [VIOM2VitalInput](#VIOM2VitalInput)[ViomId].ViomLoopHour

else:

return [LatestVIOM2LoopHourVIOM](#LatestVIOM2LoopHourVIOM)[ViomId](k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0987], [iTC\_CC-SyAD-0988]

[End]

#### Identical VIOM inputs

对于来自VIOM1或VIOM2的各2条安全输入信息消息，ATP软件需分别进行 “或”的处理，生成同步后的VIOM信息，如Table 5‑4所示，供离线生成代码使用，其详细定义见[REF11]。

ATP shall perform logical "or" operation between two VIOM messages at each cab, and generate one identical VIOM input message for each cab, as shown in Table 5‑4. The message as an interface, are available to offline generated code, defined in [REF11].

Table 5‑4 Identical VIOM Inputs for offline application

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
|  | [VIOM1VitalInputsValid](#VIOM1VitalInputsValid) | NUMERIC\_32 | 同步后的VIOM1过期时间 |
|  | [VIOM1VitalInput](#VIOM1VitalInput)[**MAX\_VITAL\_INPUT\_NB**] | BOOLEAN | 同步后的VIOM1安全输入 |
|  | [VIOM2VitalInputsValid](#VIOM2VitalInputsValid) | NUMERIC\_32 | 同步后的VIOM2过期时间 |
|  | [VIOM2VitalInput](#VIOM2VitalInput) [**MAX\_VITAL\_INPUT\_NB**] | BOOLEAN | 同步后的VIOM2安全输入 |

[iTC\_CC\_ATP-SwRS-0538]

VIOM1VitalInputsValid，判断来自**END\_1**的VIOM安全输入信息是否在有效时间内。

ATP determines whether the vital inputs message from VIOM1 valid.

def [VIOM1VitalInputsValid](#VIOM1VitalInputsValid)(k):

return (Message.[Valid](#Valid)([VIOM1VitalInput](#VIOM1VitalInput)(1).AtpLoopHour, **VIOM\_VALIDITY\_TIME**)

or Message.[Valid](#Valid)([VIOM1VitalInput](#VIOM1VitalInput)(2).AtpLoopHour, **VIOM\_VALIDITY\_TIME**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0987], [iTC\_CC-SyAD-0988], [iTC\_CC\_ATP\_SwHA-0227]

[End]

[iTC\_CC\_ATP-SwRS-0539]

VIOM2VitalInputsValid，判断来自**END\_2**的VIOM安全输入信息是否在有效时间内。

ATP determines whether the vital inputs message from VIOM2 valid.

def [VIOM2VitalInputsValid](#VIOM2VitalInputsValid)(k):

return (Message.[Valid](#Valid)([VIOM2VitalInput](#VIOM2VitalInput)(1).AtpLoopHour, **VIOM\_VALIDITY\_TIME**)

or Message.[Valid](#Valid)([VIOM2VitalInput](#VIOM2VitalInput)(2).AtpLoopHour, **VIOM\_VALIDITY\_TIME**))

#Category= Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0987], [iTC\_CC-SyAD-0988], [iTC\_CC\_ATP\_SwHA-0227]

[End]

[iTC\_CC\_ATP-SwRS-0061]

IdenticalInputs， 对于**END\_1**或**END\_2**每个车头的2个VIOM，ATP软件需根据其消息有效性，对它们取“或”，即：

* 如果同一端车头2个VIOM消息均无效，则该端IdenticalInputs无效；
* 否则，如果同一端车头只有1个VIOM消息有效，则IdenticalInputs等于该有效的VIOM消息；
* 否则，如果同一端2个VIOM至少有一个VIOM消息是“允许”状态，则ATP认为IdenticalInputs中该端VIOM的相应端口是“允许状态”。
* 否则，ATP认为IdenticalInputs中该端VIOM的相应端口是“限制”状态。

For four vital input messages from VIOM1 and VIOM2, ATP shall deal with it comprehensively and generate the [IdenticalInputs](#IdenticalInputs) information in this cycle for calculation of other modules, structured as . For both of VIOM message in the same END, ATP shall consider the reliability and use “logic OR” for judge the status of each port:

* If both of VIOM message from one train end is invalid, the [IdenticalInputs](#IdenticalInputs) of this end is also ineffective.
* If only one VIOM message from one train end is invalid, the [IdenticalInputs](#IdenticalInputs) is equal to this effective VIOM message.
* If there is at least one VIOM message of a port is on permissive, ATP shall regard the corresponding port of this train END as permissive status.
* Otherwise, ATP shall regard this port as restrictive status in the [IdenticalInputs](#IdenticalInputs).

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0007], [iTC\_CC\_ATP\_SwHA-0176], [iTC\_CC-SyAD-1431], [iTC\_CC-SyAD-1432]

[End]

对于来自车辆的输入端口信息，可以根据项目进行配置，将不同端口的采集结果通过“与”、“或”、“非”等运算，得到期望的结果，如Figure 5‑4所示。

According to the project configuration, the value from different input ports can participate in logic operation to get desired results, as shown in Figure 5‑4.



Figure 5‑4 Configurable Rolling-stock Inputs

ATP支持的所有可配置输入见Table 5‑5所示，所列变量均由离线工具根据项目配置生成，作为项目安全数据的一部分由ATP软件读取。

All configurable inputs ATP supported are shown in Table 5‑5. All these variables are generated by offline tools and as a part of the project data read by ATP during initialization.

Table 5‑5 Configurable Inputs from Rolling Stock

|  |  |
| --- | --- |
| Operation | Description |
| [AllTrainDoorsClosedAndLocked](#AllTrainDoorsClosedAndLocked) | 两侧车门关闭并锁闭。 |
| [ATCcontrolledTrain](#ATCcontrolledTrain) | CC未被旁路，可执行控车功能 |
| [BlockModeUsed](#BlockModeUsed) | BM驾驶模式选择，使用BM下的EOA |
| [BMoverlapReleasableSendable](#BMoverlapReleasableSendable) | 可通过无线向CI发送Overlap解锁消息 |
| [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid) | 使用BM下的变量 |
| [CoercedPermissive](#CoercedPermissive)[1..4] | 强制允许条件1~4 |
| [ConditionForRMlimitSpeed](#ConditionForRMlimitSpeed)[1..7] | 支持7种不同的RM限速 |
| [CoupledByEnd1](#CoupledByEnd1) | End1端连挂 |
| [CoupledByEnd2](#CoupledByEnd2) | End2端连挂 |
| [DriverInCab\_1](#DriverInCab_1) | 司机选择驾驶室1 |
| [DriverInCab\_2](#DriverInCab_2) | 司机选择驾驶室2 |
| [EmergencyHandleNotPulledEnd1](#EmergencyHandleNotPulledEnd1) | 驾驶室1端的疏散门未被拉下 |
| [EmergencyHandleNotPulledEnd2](#EmergencyHandleNotPulledEnd2) | 驾驶室2端的疏散门未被拉下 |
| [EmergencyHandleNotPulledSide](#EmergencyHandleNotPulledSide) | 列车两侧的紧急开门手柄未被拉下 |
| [InhibitControlPSDstatus](#InhibitControlPSDstatus) | 禁止监控PSD状态 |
| [InhibitControlTrainDoorsStatus](#InhibitControlTrainDoorsStatus) | 禁止监控车门状态 |
| [InhibitProtectionEvacuationInDistance](#InhibitProtectionEvacuationInDistance) | 禁止离站时的逃生监控 |
| [InhibitProtectionEvacuationWithStop](#InhibitProtectionEvacuationWithStop) | 禁止站间停车时的逃生监控 |
| [InhibitPSDopeningSupervisedByATP](#InhibitPSDopeningSupervisedByATP) | 禁止发PSD开门命令时输出PB |
| [MemorizedLocationAuthorized](#MemorizedLocationAuthorized) | 授权使用记忆定位 |
| [MotionProtectionInhibition](#MotionProtectionInhibition) | 禁止CC进行列车位置监控（仅监控RM限速） |
| [NotCoercedRestrictive](#NotCoercedRestrictive)[1..4] | 非强制限制条件1~4 |
| [NoUndetectableDanger\_1](#NoUndetectableDanger_1) | 驾驶室1端有司机监控 |
| [NoUndetectableDanger\_2](#NoUndetectableDanger_2) | 驾驶室2端有司机监控 |
| [RMRselectedDrivingMode](#RMRselectedDrivingMode) | 选择倒车模式 |
| [RouteSetNotNeededSendable](#RouteSetNotNeededSendable) | 授权可以给ZC发送RSNN信息 |
| [SignalOverrideSendable](#SignalOverrideSendable) | 授权可以给ZC发送关闭信号机信息 |
| [TrainEmergencyBrakeApplied](#TrainEmergencyBrakeApplied) | 紧急制动已施加 |
| [TrainNotCoupled](#TrainNotCoupled) | 列车没有连挂 |
| [TrainParkingBrakeApplied](#TrainParkingBrakeApplied) | 停车制动已施加 |
| [TrainUnitIntegrity](#TrainUnitIntegrity) | 列车完整性可以保证 |
| [TSRcontrolInhibition](#TSRcontrolInhibition) | 禁止TSR监控 |

#### Train coupled status management

ATP支持两车的连挂，如Figure 5‑5所示，之前独立运行的两列车，可连挂组成一列车后运行。根据与车辆的接口协议，连挂后车辆听从激活端列车的两个CC控制。例如Figure 5‑5，连挂后激活Train I的END1，则所在Train I的两个CC互为冗余控车；反之，若激活Train II的END2端，则由Train II的两个CC控车。对于非激活端的两个CC，依然照常工作，但一般无法采集车辆输入，车辆也不响应其输出。



Figure 5‑5 Train Coupled Status

除去Figure 5‑5所示是两列车的END1-END2之间连挂外，还有可能END1-END1连挂和END2-END2连挂。由于连挂方式与ATP的定位有关，因此ATP需判断车辆的连挂状态与[TrainType](#TrainType)是否一致，只有在一致状态下才能进行正常定位和运行。如果连挂状态发生变化，ATP会根据连挂状态计算新的[TrainType](#TrainType)并将其写入Dataplug，需重启后重新读入使用。

[iTC\_CC\_ATP-SwRS-0075]

CoupledByEnd1或CoupledByEnd2，列车两端连挂其他车辆。如果该项目未配置连挂输入的采集，则认为列车未与其他车连挂。其状态来自于项目可配置的列车输入采集。

[CoupledByEnd1](#CoupledByEnd1) or [CoupledByEnd2](#CoupledByEnd2) shows that both ends of train connect with other trains. If the project is not configured with the capture of coupling input, it is certain that the train does not connect with other trains.

def [CoupledByEnd1](#CoupledByEnd1)(k):

return [Offline](#Offline).[GetCoupledByEnd1](#GetCoupledByEnd1)(k)

def [CoupledByEnd2](#CoupledByEnd2)(k):

return [Offline](#Offline).[GetCoupledByEnd2](#GetCoupledByEnd2)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0193], [iTC\_CC-SyAD-0211], [iTC\_CC-SyAD-0338], [iTC\_CC-SyAD-1003], [iTC\_CC\_ATP\_SwHA-0189]

[End]

[iTC\_CC\_ATP-SwRS-0802]

TrainNotCoupled，列车未与其他车辆连挂。

def [TrainNotCoupled](#TrainNotCoupled)(k):

return [Offline](#Offline).[GetTrainNotCoupled](#GetTrainNotCoupled)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1439]

[End]

[iTC\_CC\_ATP-SwRS-0789]

TrainCoupledType，根据项目配置，获取当前列车的连挂类型。支持以下四种连挂类型：

* **TRAIN\_COUPLED\_UNKNOWN**，当前连挂状态无效；
* **TRAIN\_NO\_COUPLED**，列车未连挂；
* **TRAIN\_COUPLED\_END1**，列车END\_1端连挂；
* **TRAIN\_COUPLED\_END2**，列车END\_2端连挂。

规则如下：

def [TrainCoupledType](#TrainCoupledType)(k):

if ([TrainNotCoupled](#TrainNotCoupled)(k)

and not [TrainCoupledByEnd1](#TrainCoupledByEnd1)(k)

and not [TrainCoupledByEnd2](#TrainCoupledByEnd2)(k)):

return **TRAIN\_NO\_COUPLED**

elif (not [TrainNotCoupled](#TrainNotCoupled)(k)

and [TrainCoupledByEnd1](#TrainCoupledByEnd1)(k)

and not [TrainCoupledByEnd2](#TrainCoupledByEnd2)(k)):

return **TRAIN\_COUPLED\_END1**

elif (not [TrainNotCoupled](#TrainNotCoupled)(k)

and not [TrainCoupledByEnd1](#TrainCoupledByEnd1)(k)

and [TrainCoupledByEnd2](#TrainCoupledByEnd2)(k)):

return **TRAIN\_COUPLED\_END2**

else:

return **TRAIN\_COUPLED\_UNKNOWN**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1439]

[End]

**NOTES:**

配置数据中的[TrainNotCoupled](#TrainNotCoupled)，[TrainCoupledByEnd1](#TrainCoupledByEnd1)和[TrainCoupledByEnd2](#TrainCoupledByEnd2)应考虑VIOM消息的有效性

* 对于单头CC就能采集两端驾驶室连挂状态的项目，两端的VIOM消息中至少有一个有效即可；
* 对于单头CC仅采集本端连挂状态的项目，两端VIOM消息必须全有效，才能判断连挂状态有效。

当不满足上述条件，即VIOM消息无效时，[TrainCoupledType](#TrainCoupledType)应当是**TRAIN\_COUPLED\_UNKNOWN**状态。

[iTC\_CC\_ATP-SwRS-0790]

CoupledTypeInconsistent，比较列车类型与当前列车的连挂类型是否一致。

* 初始化时，该值为**False**；
* 判断列车类型[TrainType](#TrainType)与连挂类型[TrainCoupledType](#TrainCoupledType)是否一致：
* 若不一致，则设置该值为**True**；
* 否则，设置该值为**False**
* 其他情况，保持不变。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1440]

[End]

[iTC\_CC\_ATP-SwRS-0792]

MatchRebootCondition，是否由于连挂类型改变而设置ATP重启。

* 初始化时，[MatchRebootCondition](#MatchRebootCondition)为**False**；
* 当满足以下条件后，ATP将计算新的列车类型[TrainType](#TrainType)并写入Dataplug，同时设置[MatchRebootCondition](#MatchRebootCondition)为**True**，请求重启ATP。
* [CoupledTypeInconsistent](#CoupledTypeInconsistent)为**True**；
* 且[TrainCoupledType](#TrainCoupledType)不是**TRAIN\_COUPLED\_UNKNOWN**；
* 且列车处于[TrainFilteredStopped](#TrainFilteredStopped)状态。
* 其他情况，[MatchRebootCondition](#MatchRebootCondition)保持不变。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1440]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [CoupledTypeInconsistent](#CoupledTypeInconsistent) | √ | √ | BOOLEAN |
| [IdenticalInputs](#IdenticalInputs) | √ | √ | BOOLEAN |
| [LatestVIOM1LoopHourVIOM](#LatestVIOM1LoopHourVIOM) | √ | √ | NUMERIC\_32 |
| [LatestVIOM2LoopHourVIOM](#LatestVIOM2LoopHourVIOM) | √ | √ | NUMERIC\_32 |
| [MatchRebootCondition](#MatchRebootCondition) | √ | √ | BOOLEAN |
| [TrainCoupledType](#TrainCoupledType) | √ | √ | NUMERIC\_32 |

## F13-Manage Loop Hour with Distant ATP

本模块处理来自冗余ATP的信息，并维护ATP所使用的loop hour，作为判断与外部系统通信时间有效性的依据。

This module handles information from the redundant ATP, and generates loop hour as the basis of validity judgment.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [CoreId](#CoreId) | Internal | F11-Acquire Configuration Data |
| [OtherCCsynchroReport](#OtherCCsynchroReport) | External | Interface with Distant ATP |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [LastOtherATPmessageAge](#LastOtherATPmessageAge) | √ | √ | NUMERIC\_32 |
| [OtherATPmessageAvailable](#OtherATPmessageAvailable) | √ | √ | BOOLEAN |
| [OtherATPmessageReceived](#OtherATPmessageReceived) | √ | √ | BOOLEAN |

#### ProcessingLoop hour management

ATP从上电开始，需根据自己的周期数和所在的列车车头，维护自身的loop hour，作为与外界通信的时间标签，用于监控通信的时效性。对于两端驾驶室的ATP所使用的初始以及最大时间均不同，且没有交集，即根据消息中的loop hour，也能分辨出该消息是来自或者发往哪一端驾驶室的ATP。

Since power up, ATP shall maintain its loop hour as a label used to monitor the timeline of communication, according to the cycle number and the cab where ATP settled. For both ends of the cab of the ATP, they use different loop hour initial value, and there is no intersection between the ranges. Thus, according to the message loop hour, the source of the message sent from which ATP can distinguish.

[iTC\_CC\_ATP-SwRS-0144]

ATPtime，维护本端ATP的loop hour时间。

* 根据本端[CoreId](#CoreId)，初始化为**END\_1**或 **END\_2**的初始值；
* 如果超过了相应的最大值，则重新等于初始化的值。
* 否则每周期加1

[ATPtime](#ATPtime) stands for the ATP loop hour of this train END.

* Based on [CoreId](#CoreId), ATP initialize [ATPtime](#ATPtime) as the initiative value of **END\_1** or **END\_2**;
* If the value exceeds the maximum loop hour, ATP shall set it as the initiative value;
* Otherwise, add one for each cycle.

def [ATPtime](#ATPtime)(k):

if ([CoreId](#CoreId)(k) is **END\_1**):

if (Initialization):

return **CC1\_INIT\_TIME**

elif ([ATPtime](#ATPtime)(k-1) >= **CC1\_MAX\_TIME**):

return **CC1\_INIT\_TIME**

else:

return [ATPtime](#ATPtime)(k-1) + 1

else:

if (Initialization):

return **CC2\_INIT\_TIME**

elif ([ATPtime](#ATPtime)(k-1) >= **CC2\_MAX\_TIME**):

return **CC2\_INIT\_TIME**

else:

return [ATPtime](#ATPtime)(k-1) + 1

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0208], [iTC\_CC-SyAD-0209], [iTC\_CC-SyAD-0221], [iTC\_CC\_ATP\_SwHA-0016]

[End]

#### loop hour management for the other ATP

ATP维护位于另一端车头的冗余ATP的时间，用于监控外部系统发给冗余端ATP消息的有效性。考虑到消息传输延迟，该冗余ATP时间应当在最大最小值范围内。

The ATP software needs to maintain the redundant ATP cycle time for monitoring the effectiveness of the message sent to the redundant side. Taking into account the transmission delay time, the redundant ATP time should be within the maximum and minimum range.

[iTC\_CC\_ATP-SwRS-0615]

OtherATPmessageReceived，本周期收到冗余ATP消息并校验正确。

The message transmitted from the distant ATP in the other END shall be protected by check words. And before using the information, ATP shall verify the check words.

def [OtherATPmessageReceived](#OtherATPmessageReceived)(k):

return [Message.Received](#Received)([OtherCCsynchroReport](#OtherCCsynchroReport),k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0967], [iTC\_CC\_ATP\_SwHA-0243]

[End]

[iTC\_CC\_ATP-SwRS-0078]

OtherATPmessageAvailable，判断来自冗余ATP消息的有效性：

[OtherATPmessageAvailable](#OtherATPmessageAvailable), ATP shall judge the effectiveness of message from the redundant ATP, shown as following pseudo-codes:

def [OtherATPmessageAvailable](#OtherATPmessageAvailable)(k):

return Message.[Available](#Available)([OtherATPmessageReceived](#OtherATPmessageReceived)(k),

[OtherCCsynchroReport](#OtherCCsynchroReport).LatestTimeOtherCore,

**OTHER\_ATP\_VALIDITY\_TIME**,

[LastOtherATPmessageAge](#LastOtherATPmessageAge)(k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0967], [iTC\_CC\_ATP\_SwHA-0013], [iTC\_CC\_ATP\_SwHA-0014]

[End]

[iTC\_CC\_ATP-SwRS-0616]

LastOtherATPmessageAge，获取到的远端ATP消息的存活时间。

def [LastOtherATPmessageAge](#LastOtherATPmessageAge)(k):

return Message.[LastAge](#LastAge)([OtherATPmessageAvailable](#OtherATPmessageAvailable)(k),

[OtherCCsynchroReport](#OtherCCsynchroReport).LatestTimeOtherCore,

[LastOtherATPmessageAge](#LastOtherATPmessageAge)(k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0967]

[End]

[iTC\_CC\_ATP-SwRS-0540]

OtherATPmessageValid，接收到的冗余ATP消息是否在有效期内。如果该消息已失效，则设置OtherATPmessageValid为**False**；否则为**True**。

[OtherATPmessageValid](#OtherATPmessageValid) represents the effectiveness of the messages from redundant ATP. If this message is invalid, ATP will set [OtherATPmessageValid](#OtherATPmessageValid) as **False**; otherwise, it is set as **True**.

def [OtherATPmessageValid](#OtherATPmessageValid)(k):

return Message.[Valid](#Valid)([OtherCCsynchroReport](#OtherCCsynchroReport).LatestTimeOtherCore,

**OTHER\_ATP\_VALIDITY\_TIME**,

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0967]

[End]

[iTC\_CC\_ATP-SwRS-0081]

OtherATPminTime，本端ATP维护的冗余ATP的最小时间。设置规则如下：

* 初始化时根据所在车头设置OtherATPminTime为默认值；
* 否则，如果本周期收新的冗余ATP消息可用，则更新OtherATPminTime为消息中的currentTime；
* 否则，如果冗余ATP消息仍然在有效期内，则对OtherATPminTime每周期加1，若越界则重新等于初始化值；
* 否则，如果当前收到的新的冗余ATP消息（但不可用），则将OtherATPminTime更新为消息中的时间；
* 其他情况，OtherATPminTime累加1，若越界则重新等于初始化值。

The [OtherATPminTime](#OtherATPminTime) stands for the local ATP maintained minimum time of the redundant ATP. The setting rule is as following:

* In initialization, ATP set the [OtherATPminTime](#OtherATPminTime) as default value based on the [CoreId](#CoreId) of the redundant ATP.
* Or else:, if the updating message from the new redundant ATP in this cycle is available, ATP will update [OtherATPminTime](#OtherATPminTime) as the current time in the message.
* Or else:, if the redundant ATP message is still effective, ATP will add 1 in the [OtherATPminTime](#OtherATPminTime) until it is out of bound, and set is as initialization value.
* Or else:, If the received a new redundant ATP message, but it was not available, ATP shall update [OtherATPminTime](#OtherATPminTime) as in the message.
* Otherwise, accumulate [OtherATPminTime](#OtherATPminTime).

if ([CoreId](#CoreId) == **END\_1**)

if (Initialization)

[OtherATPminTime](#OtherATPminTime) = **CC2\_INIT\_TIME**

elif (([OtherATPmessageAvailable](#OtherATPmessageAvailable)(k) == **True**)

or (([OtherATPmessageValid](#OtherATPmessageValid)(k) == **False**)

and [Exists](#Exists)([OtherCCsynchroReport](#OtherCCsynchroReport))))

[OtherATPminTime](#OtherATPminTime) = [OtherCCsynchroReport](#OtherCCsynchroReport).CurrentTime

else:

if ([OtherATPminTime](#OtherATPminTime)(k-1) >= **CC2\_MAX\_TIME**)

[OtherATPminTime](#OtherATPminTime) = **CC2\_INIT\_TIME**

else:

[OtherATPminTime](#OtherATPminTime) = [OtherATPminTime](#OtherATPminTime)(k-1) + 1

else:

if (Initialization)

[OtherATPminTime](#OtherATPminTime) = **CC1\_INIT\_TIME**

elif (([OtherATPmessageAvailable](#OtherATPmessageAvailable)(k) == **True**)

or (([OtherATPmessageValid](#OtherATPmessageValid)(k) == **False**)

and [Exists](#Exists)([OtherCCsynchroReport](#OtherCCsynchroReport))))

[OtherATPminTime](#OtherATPminTime) = [OtherCCsynchroReport](#OtherCCsynchroReport).CurrentTime

else:

if ([OtherATPminTime](#OtherATPminTime)(k-1) >= **CC1\_MAX\_TIME**)

[OtherATPminTime](#OtherATPminTime) = **CC1\_INIT\_TIME**

else:

[OtherATPminTime](#OtherATPminTime) = [OtherATPminTime](#OtherATPminTime)(k-1) + 1

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0962]

[End]

[iTC\_CC\_ATP-SwRS-0083]

OtherATPmaxTime，维护冗余ATP的最大时间。

The [OtherATPmaxTime](#OtherATPmaxTime) stands for the local ATP maintained maximum time of the redundant ATP. The rules to update [OtherATPmaxTime](#OtherATPmaxTime) are similar with [OtherATPminTime](#OtherATPminTime) except that when received a new message from the redundant ATP, the [OtherATPmaxTime](#OtherATPmaxTime) shall add the maximum transmission delay in network.

if ([CoreId](#CoreId) == **END\_1**)

if (Initialization)

[OtherATPmaxTime](#OtherATPmaxTime) = **CC2\_INIT\_TIME**

elif (([OtherATPmessageAvailable](#OtherATPmessageAvailable)(k) == **True**)

or (([OtherATPmessageValid](#OtherATPmessageValid)(k) == **False**)

and [Exists](#Exists)([OtherCCsynchroReport](#OtherCCsynchroReport))))

[OtherATPmaxTime](#OtherATPmaxTime) = [OtherCCsynchroReport](#OtherCCsynchroReport).CurrentTime

+ LoopHourModularSub(ATPtime(k),

[OtherCCsynchroReport](#OtherCCsynchroReport).LatestTimeOtherCore)

else:

if ([OtherATPmaxTime](#OtherATPmaxTime)(k-1) >= **CC2\_MAX\_TIME**)

[OtherATPmaxTime](#OtherATPmaxTime) = **CC2\_INIT\_TIME**

else:

[OtherATPmaxTime](#OtherATPmaxTime) = [OtherATPmaxTime](#OtherATPmaxTime)(k-1) + 1

else:

if (Initialization)

[OtherATPmaxTime](#OtherATPmaxTime) = **CC1\_INIT\_TIME**

elif (([OtherATPmessageAvailable](#OtherATPmessageAvailable)(k) == **True**)

or (([OtherATPmessageValid](#OtherATPmessageValid)(k) == **False**)

and [Exists](#Exists)([OtherCCsynchroReport](#OtherCCsynchroReport))))

[OtherATPmaxTime](#OtherATPmaxTime) = [OtherCCsynchroReport](#OtherCCsynchroReport).CurrentTime

+ LoopHourModularSub(ATPtime(k),

[OtherCCsynchroReport](#OtherCCsynchroReport).LatestTimeOtherCore)

else:

if ([OtherATPmaxTime](#OtherATPmaxTime)(k-1) >= **CC1\_MAX\_TIME**)

[OtherATPmaxTime](#OtherATPmaxTime) = **CC1\_INIT\_TIME**

else:

[OtherATPmaxTime](#OtherATPmaxTime) = [OtherATPmaxTime](#OtherATPmaxTime)(k-1) + 1

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0962]

[End]

#### Parse distant ATP information

[iTC\_CC\_ATP-SwRS-0080]

OtherATP，解析并存储远端ATP的消息。

* 初始化或者远端消息过期时，设置相应的值为默认状态；
* 当本周期收到新的远端消息时，将其设置为新收到消息的值；
* 否则，保持不变

OtherATP, parse and store messages from the distant ATP.

* In initialization or the message has expired, set all variables as default value;
* when new message available, set the corresponding value from the new message;
* otherwise, remain unchanged.

def [OtherATP](#OtherATP)(k):

if (Initialization

or (not [OtherATPmessageValid](#OtherATPmessageValid)(k))):

[OtherATP](#OtherATP).LatestTimeOtherCore = **INVALID\_LOOP\_HOUR**

[OtherATP](#OtherATP).CoreId = **None**

[OtherATP](#OtherATP).BeaconId = **None**

[OtherATP](#OtherATP).EnableDoorOpening\_A = **False**

[OtherATP](#OtherATP).EnableDoorOpening\_B = **False**

[OtherATP](#OtherATP).PsdManagerOpeningOrder = **False**

[OtherATP](#OtherATP).PsdIdSide\_A = **None**

[OtherATP](#OtherATP).PsdValiditySide\_A = **None**

[OtherATP](#OtherATP).PsdClosedSide\_A = **False**

[OtherATP](#OtherATP).PsdIdSide\_B = **None**

[OtherATP](#OtherATP).PsdValiditySide\_B = **None**

[OtherATP](#OtherATP).PsdClosedSide\_B = **False**

[OtherATP](#OtherATP).ZcVersion = **None**

[OtherATP](#OtherATP).LocatedOnKnownPath = **False**

[OtherATP](#OtherATP).LocatedWithMemLocation = **False**

[OtherATP](#OtherATP).Location.Ext2 = **None**

[OtherATP](#OtherATP).Location.Uncertainty = **None**

[OtherATP](#OtherATP).Location.Ext1 = **None**

[OtherATP](#OtherATP).SleepZoneId = **None**

[OtherATP](#OtherATP).SleepZoneVersion = **None**

[OtherATP](#OtherATP).MotionSinceLastReloc = **None**

[OtherATP](#OtherATP).MotionSinceMemLoc = **None**

[OtherATP](#OtherATP).TrainFilteredStopped = **False**

[OtherATP](#OtherATP).SafetyParameterVersion = **None**

[OtherATP](#OtherATP).SafetyApplicationVersion = **None**

[OtherATP](#OtherATP).CC\_SSID = **None**

[OtherATP](#OtherATP).OverlapExpired = **False**

elif ([OtherATPmessageAvailable](#OtherATPmessageAvailable)(k)):

[OtherATP](#OtherATP).LatestTimeOtherCore = [OtherCCsynchroReport](#OtherCCsynchroReport).LatestTimeOtherCore(k)

[OtherATP](#OtherATP).CoreId = [OtherCCsynchroReport](#OtherCCsynchroReport).CoreId

[OtherATP](#OtherATP).BeaconId = [OtherCCsynchroReport](#OtherCCsynchroReport).BeaconId

[OtherATP](#OtherATP).EnableDoorOpening\_A = [OtherCCsynchroReport](#OtherCCsynchroReport).EnableDoorOpening\_A

[OtherATP](#OtherATP).EnableDoorOpening\_B = [OtherCCsynchroReport](#OtherCCsynchroReport).EnableDoorOpening\_B

[OtherATP](#OtherATP).PsdManagerOpeningOrder = [OtherCCsynchroReport](#OtherCCsynchroReport).PsdManagerOpeningOrder

[OtherATP](#OtherATP).PsdIdSide\_A = [OtherCCsynchroReport](#OtherCCsynchroReport).PsdIdSide\_A

[OtherATP](#OtherATP).PsdValiditySide\_A = [OtherCCsynchroReport](#OtherCCsynchroReport).PsdValiditySide\_A

[OtherATP](#OtherATP).PsdClosedSide\_A = [OtherCCsynchroReport](#OtherCCsynchroReport).PsdClosedSide\_A

[OtherATP](#OtherATP).PsdIdSide\_B = [OtherCCsynchroReport](#OtherCCsynchroReport).PsdIdSide\_B

[OtherATP](#OtherATP).PsdValiditySide\_B = [OtherCCsynchroReport](#OtherCCsynchroReport).PsdValiditySide\_B

[OtherATP](#OtherATP).PsdClosedSide\_B = [OtherCCsynchroReport](#OtherCCsynchroReport).PsdClosedSide\_B

[OtherATP](#OtherATP).ZcVersion = [OtherCCsynchroReport](#OtherCCsynchroReport).ZcVersion

[OtherATP](#OtherATP).LocatedOnKnownPath = [OtherCCsynchroReport](#OtherCCsynchroReport).LocatedOnKnownPath

[OtherATP](#OtherATP).LocatedWithMemLocation = [OtherCCsynchroReport](#OtherCCsynchroReport).LocatedWithMemLocation

[OtherATP](#OtherATP).Location.Ext2 = [OtherCCsynchroReport](#OtherCCsynchroReport).Location.Ext2

[OtherATP](#OtherATP).Location.Uncertainty = [OtherCCsynchroReport](#OtherCCsynchroReport).Location.Uncertainty

[OtherATP](#OtherATP).Location.Ext1 = [OtherCCsynchroReport](#OtherCCsynchroReport).Location.Ext1

[OtherATP](#OtherATP).SleepZoneId = [OtherCCsynchroReport](#OtherCCsynchroReport).SleepZoneId

[OtherATP](#OtherATP).SleepZoneVersion = [OtherCCsynchroReport](#OtherCCsynchroReport).SleepZoneVersion

[OtherATP](#OtherATP).MotionSinceLastReloc = [OtherCCsynchroReport](#OtherCCsynchroReport).MotionSinceLastReloc

[OtherATP](#OtherATP).MotionSinceMemLoc = [OtherCCsynchroReport](#OtherCCsynchroReport).MotionSinceMemLoc

[OtherATP](#OtherATP).TrainFilteredStopped = [OtherCCsynchroReport](#OtherCCsynchroReport).TrainFilteredStopped

[OtherATP](#OtherATP).SafetyParameterVersion = [OtherCCsynchroReport](#OtherCCsynchroReport).SafetyParameterVersion

[OtherATP](#OtherATP).SafetyApplicationVersion = [OtherCCsynchroReport](#OtherCCsynchroReport).SafetyApplicationVersion

[OtherATP](#OtherATP).CC\_SSID = [OtherCCsynchroReport](#OtherCCsynchroReport).CC\_SSID

[OtherATP](#OtherATP).OverlapExpired = [OtherCCsynchroReport](#OtherCCsynchroReport).OverlapExpired

else:

pass

return [OtherATP](#OtherATP)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0963], [iTC\_CC-SyAD-1212], [iTC\_CC\_ATP\_SwHA-0014], [iTC\_CC\_ATP\_SwHA-0013]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [ATPtime](#ATPtime) | √ | √ | NUMERIC\_32 |
| [OtherATP](#OtherATP) | √ | √ | ST\_SYNCHRO\_REPORT |
| [OtherATPmaxTime](#OtherATPmaxTime) | √ | √ | NUMERIC\_32 |
| [OtherATPmessageValid](#OtherATPmessageValid) | √ | √ | BOOLEAN |
| [OtherATPminTime](#OtherATPminTime) | √ | √ | NUMERIC\_32 |

## F14-Manage Variants in Block Mode

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [ATPtime](#ATPtime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [BeaconMessage](#BeaconMessage) | Internal | F21-Manage Interface with |
| [BeaconMessageReceive](#BeaconMessageReceive) | Internal | F21-Manage Interface with |
| [CBIvariantReport](#CBIvariantReport) | Internal | F41-Determine the EOA |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [OtherATPmaxTime](#OtherATPmaxTime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainFrontOrientation](#TrainFrontOrientation) | Internal | F33-Confirm Train Localization |
| [TrainLocalized](#TrainLocalized) | Internal | F32-Update Train Location |
| [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath) | Internal | F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [BMbeaconVariantValue](#BMbeaconVariantValue) | √ | × | BOOLEAN |
| [BMbeaconVariants](#BMbeaconVariants) | √ | √ | BOOLEAN |
| [BMcbiVariantValue](#BMcbiVariantValue) | √ | × | BOOLEAN |
| [BMcbiVariants](#BMcbiVariants) | √ | √ | BOOLEAN |
| [BMvariantValidLastRisingAge](#BMvariantValidLastRisingAge) | √ | √ | NUMERIC\_32 |
| [BeaconVariantsUpdating](#BeaconVariantsUpdating) | √ | √ | BOOLEAN |
| [CBIvariantAge](#CBIvariantAge) | √ | √ | NUMERIC\_32 |
| [CBIvariantReportAvailable](#CBIvariantReportAvailable) | √ | √ | BOOLEAN |
| [CBIvariantReportReceived](#CBIvariantReportReceived) | √ | √ | BOOLEAN |
| [UsedBMbeaconId](#UsedBMbeaconId) | √ | √ | NUMERIC\_32 |

### Processing

本模块管理Block模式下使用变量，包括使用的是来自BM信标的变量还是与CI无线通信的变量。

This module mange the variants used in block mode, including the BM beacon and the CBI radio.

#### Determine block mode

[iTC\_CC\_ATP-SwRS-0067]

BlockModeUsed，当前是否现在选择BM模式。其状态来自于项目可配置的列车输入采集。

[BlockModeUsed](#BlockModeUsed) represents that either of train end chooses BM mode.

def [BlockModeUsed](#BlockModeUsed)(k):

return [Offline.GetBlockModeUsed](#GetBlockModeUsed)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0948], [iTC\_CC-SyAD-1003], [iTC\_CC\_ATP\_SwHA-0203]

[End]

[iTC\_CC\_ATP-SwRS-0066]

BMvariantValidWhileTemporallyValid，当前是否使用BM变量。其状态来自于项目可配置的列车输入采集。

The status of [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid) shows whether it is in the BM mode.

def [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)(k):

return [Offline.GetBMvariantValidWhileTemporallyValid](#GetBMvariantValidWhileTemporallyValid)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0298], [iTC\_CC-SyAD-0841], [iTC\_CC-SyAD-1003], [iTC\_CC\_ATP\_SwHA-0202]

[End]

#### BM variants received from beacon

在BLOCK运营模式下，线路上的变量信息来自有源信标消息的解析。对于每个有源信标，有其特定的方向，且最多存储16个变量的状态。对于每个变量所代表的线路设备的含义及其有效期，在离线数据中进行配置。在BLOCK运营模式下，ATP仅存储与列车运营方向相同最新的一个BM信标的变量。

In the block mode, the variants come from the parsing of BM beacon message. For each BM beacon, has a specific orientation and can store 16 variants at most. For the content and validity of each variant is defined in the off-line data. During the block mode, ATP shall only store the last read BM variants with the same direction as the train moving.

[iTC\_CC\_ATP-SwRS-0146]

BeaconVariantsUpdating，判断是否要更新BM变量。

* 若本周期满足以下所有条件时，则认为需要更新BM变量，设置BeaconVariantsUpdating为**True**。
* 当前使用BM变量（[BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)）；
* 本周期未停车且收到信标消息且判断该信标带有BM变量；
* 上周期列车未定位，或该BM信标方向与列车运营方向一致。
* 否则，设置BeaconVariantsUpdating为**False**。

[BeaconVariantsUpdating](#BeaconVariantsUpdating) used to determine ATP whether to update the BM variants in this cycle.

* If all the following conditions are fulfilled, ATP shall set [BeaconVariantsUpdating](#BeaconVariantsUpdating) as **True**:
* The current operational mode is BLOCK MODE;
* And train moved and ATP received a BM beacon in this cycle;
* And the train is either not localized, or the direction of the BM variants is as same as the orientation of the train front end.
* Otherwise, ATP shall set [BeaconVariantsUpdating](#BeaconVariantsUpdating) as **False**.

def [BeaconVariantsUpdating](#BeaconVariantsUpdating)(k):

return ([BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)(k)

and [**BeaconMessageReceive**](#BeaconMessageReceive)(k)

and [TrackMap](#TrackMap).[IsBmBeacon](#IsBmBeacon)([**BeaconMessage**](#BeaconMessage).ID)

and not [TrainFilteredStopped](#TrainFilteredStopped)(k)

and (not [TrainLocalized](#TrainLocalized)(k-1)

or ([**TrackMap**](#TrackMap).[BmBeaconDirection](#BmBeaconDirection)([**BeaconMessage**](#BeaconMessage).ID)== [TrainFrontOrientation](#TrainFrontOrientation)(k-1))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0166], [iTC\_CC-SyAD-0298], [iTC\_CC-SyAD-0299], [iTC\_CC\_ATP\_SwHA-0040]

[End]

用于比较来自信标的BM变量与来自无线的BM变量哪个更新。

[iTC\_CC\_ATP-SwRS-0617]

BMbeaconReadAge，记录读取BM信标到当前的时间，默认值为**REPORT\_AGE\_MAX**。

* 如果BM信标变量无效，该值应被设置为默认值，BM信标变量无效的条件如下：
* 初始化；
* 或当前不在BM模式(not [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid))；
* 或[BMbeaconReadAge](#BMbeaconReadAge)已大于[ATPsetting](#ATPsetting).VariantsBMfullValidityTime；
* 或本周期收到的BM信标（[BeaconVariantsUpdating](#BeaconVariantsUpdating)为**True**）中[DefaultMessage](#DefaultMessage)为**True**或[BlockModeVariantAvailable](#BlockModeVariantAvailable)为**False**；
* 或本周期列车由定位转为失位状态；
* 或当前使用的BM信标方向与已定位的列车运营方向[TrainFrontOrientation](#TrainFrontOrientation)不同。
* 否则，如果本周期更新BM信标，则将该变量的初始值设置为1（因为ATP使用的是上个周期读到的信标信息）。
* 其他情况，累加该变量。

def [BMbeaconReadAge](#BMbeaconReadAge)(k):

if (Initialization

or not [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)(k)

or [BMbeaconReadAge](#BMbeaconReadAge)(k-1) > ([ATPsetting](#ATPsetting).VariantsBMfullValidityTime - 1)

or ([BeaconVariantsUpdating](#BeaconVariantsUpdating)(k)

and ([DefaultMessage](#DefaultMessage)(k) or not [BlockModeVariantAvailable](#BlockModeVariantAvailable)(k)))

or ([TrainLocalized](#TrainLocalized)(k-1)

and (not [TrainLocalized](#TrainLocalized)(k)

or TrackMap.BmBeaconDirection([UsedBMbeaconId](#UsedBMbeaconId)(k-1))

is not [TrainFrontOrientation](#TrainFrontOrientation)(k-1)))):

return **REPORT\_AGE\_MAX**

elif ([BeaconVariantsUpdating](#BeaconVariantsUpdating)(k)):

return 1

else:

return [BMbeaconReadAge](#BMbeaconReadAge)(k-1) + 1

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0168]

[End]

[iTC\_CC\_ATP-SwRS-0147]

BMbeaconVariants[**MAX\_BM\_VARIANT\_NB**]，存储BLOCK模式下的变量，每个数组元素的结构为ST\_BM\_VARIANT.

更新规则如下：

* 如果[BMbeaconReadAge](#BMbeaconReadAge)大于[ATPsetting](#ATPsetting).VariantsBMfullValidityTime（即为默认值），则设置所有变量BMbeaconVariants为限制状态，认为BM变量无效；
* 否则，如果本周期[BeaconVariantsUpdating](#BeaconVariantsUpdating)为**True**，则根据线路地图中相应的BM信标，更新每个变量的ValidityTime，LineSection和Index，并使用[BeaconMessage](#BeaconMessage).Variants更新变量状态。对于未在该信标中更新的变量，应设置为限制状态。
* 否则，BM信标变量保持不变。

The structure of array ATP stored [BMbeaconVariants](#BMbeaconVariants) are ST\_BM\_VARIANT. The rules to update the BM variants are as follows:

* if the [BMbeaconReadAge](#BMbeaconReadAge) is larger than the [ATPsetting](#ATPsetting).VariantsBMfullValidityTime, ATP shall set all BM variants as restricted status.
* Else If the [BeaconVariantsUpdating](#BeaconVariantsUpdating) is **True**, then ATP update the [BMbeaconVariants](#BMbeaconVariants) by the new beacon.
* Otherwise, keep BMbeaconVariants unchanged.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0163], [iTC\_CC-SyAD-0166], [iTC\_CC-SyAD-0167], [iTC\_CC-SyAD-0168], [iTC\_CC-SyAD-0169], [iTC\_CC-SyAD-0170], [iTC\_CC-SyAD-0297], [iTC\_CC-SyAD-0299], [iTC\_CC-SyAD-0841], [iTC\_CC\_ATP\_SwHA-0042], [iTC\_CC\_ATP\_SwHA-0043]

[End]

**NOTES**:

假设某BM信标是ATP定位使用的第二个信标，而该BM信标的变量方向与列车实际定位方向相反。由于软件功能模块执行顺序的原因，当ATP获取该BM信标的变量信息时，可能还未判断出列车行驶方向（此时尚未执行到定位模块），因此仍然会存储该信标中的变量及其更新有效期。但在执行EOA计算时，会按照列车运行方向（此时已执行完成了定位模块）向下游搜索限制点。但由于存储的变量方向不同，所有列车运行方向下游的带变量奇点均为限制状态，因此上述处理不会影响安全。当下一个周期，ATP发现该BM信标的变量方向与运行方向不符，将其清除。

There is a situation that ATP read one BM beacon as the second beacon for ATP initializing location, and the direction of this BM beacon is opposite with the train movement.

[iTC\_CC\_ATP-SwRS-0618]

BMbeaconVariantValue，获取来自BM信标中该变量的值，输入索引和周期，若过期为假值

def [BMbeaconVariantValue](#BMbeaconVariantValue)(lineSection, VarIndex, k):

if ([BMbeaconReadAge](#BMbeaconReadAge)(k) > [ATPsetting](#ATPsetting).VariantsBMfullValidityTime):

return **False**

else:

for Var in [range](#range)(0, MAX\_BM\_VARIANT\_NB):

if ([BMbeaconVariants](#BMbeaconVariants)[Var].LineSection == LineSection

and [BMbeaconVariants](#BMbeaconVariants)[Var].Index == VarIndex):

return [BMbeaconVariants](#BMbeaconVariants)[Var].Value

else:

continue

else:

return **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0295]

[End]

[iTC\_CC\_ATP-SwRS-0148]

UsedBMbeaconId用于记录当前所使用的BM变量来自哪个BM信标，判断条件如下：

* 当初始化，非使用BM变量（not [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)），该信标方向与当前车头方向不符，或列车失位时，清除UsedBMbeaconId；
* 否则，如果收到有效的BM信标，记录该信标id到UsedBMbeaconId；
* 否则，保持UsedBMbeaconId不变。

[UsedBMbeaconId](#UsedBMbeaconId) records the used BM variants came from which BM beacon:

* When one of the following conditions fulfilled, ATP clear the [UsedBMbeaconId](#UsedBMbeaconId):
* initialization,
* the BLOCK MODE variant is not temporally valid,
* the direction of the used BM beacon is not as same as train front orientation,
* the train is not localized.
* Or else:, when received a valid BM beacon, ATP update [UsedBMbeaconId](#UsedBMbeaconId);
* Otherwise, keep this value unchanged.

def [UsedBMbeaconId](#UsedBMbeaconId)(k):

if ([BeaconVariantsUpdating](#BeaconVariantsUpdating)(k)):

return [BeaconMessage](#BeaconMessage).Id

elif (Initialization

or not [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)(k)

or ([**TrainLocalized**](#TrainLocalized)(k-1)

and ([**TrackMap**](#TrackMap).[BmBeaconDirection](#BmBeaconDirection)([UsedBMbeaconId](#UsedBMbeaconId)(k-1))

is not [TrainFrontOrientation](#TrainFrontOrientation)(k-1)))

or ([**TrainLocalized**](#TrainLocalized)(k-1) and not [TrainLocalized](#TrainLocalized)(k))):

return **None**

else:

return [UsedBMbeaconId](#UsedBMbeaconId)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0163]

[End]

#### BM variants received from CBI

在有无线通信的增强型后备模式下，ATP可以与联锁通过SACEM协议通信，直接获取该联锁管辖的设备变量状态：

1. 对于联锁发送的变量与ZC发送的变量，在[ATPsetting](#ATPsetting)中定义有对应关系，可以通过联锁变量索引ZC变量。
2. ATP应管理联锁的变量状态，维护其时间有效性。联锁变量的时间有效期分为两种：一种是长有效期，用于信号机和道岔位置的状态判断；一种是短有效期，用于保护区状态的判断。

[iTC\_CC\_ATP-SwRS-0619]

BMvariantValidLastRisingAge, 记录从选择BM模式到当前经过的时间

def [BMvariantValidLastRisingAge](#BMvariantValidLastRisingAge)(k):

if (not [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)(k)):

[BMvariantValidLastRisingAge](#BMvariantValidLastRisingAge) = 0

else:

[BMvariantValidLastRisingAge](#BMvariantValidLastRisingAge) = [BMvariantValidLastRisingAge](#BMvariantValidLastRisingAge)(k-1) + 1

如果未选择BM模式，则不允许使用来自CBI的变量，因此应设置初值为0，使得不可能有来自CI消息的age小于0

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1175]

[End]

[iTC\_CC\_ATP-SwRS-0620]

CBIvariantReportReceived，ATP软件收到CCNV转发的“CBI variant report”消息，并安全校核字校验正确。

def [CBIvariantReportReceived](#CBIvariantReportReceived)(cbi, k):

return [Message.Received](#Received)([CBIvariantReport](#CBIvariantReport)(cbi), k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1171]

[End]

[iTC\_CC\_ATP-SwRS-0621]

CBIvariantReportAvailable，联锁消息可用

def [CBIvariantReportAvailable](#CBIvariantReportAvailable)(cbi, k):

return Message.[Available](#Available)([CBIvariantReportReceived](#CBIvariantReportReceived)(cbi, k),

[CBIvariantReport](#CBIvariantReport)(cbi).CcLoopHour,

[ATPsetting](#ATPsetting).VariantsBMlowValidityTime,

[min](#min)([CBIvariantReportLastAge](#CBIvariantReportLastAge)(cbi, k-1),

[CBIminProductionAge](#CBIminProductionAge)(cbi, k-1),

[BMvariantValidLastRisingAge](#BMvariantValidLastRisingAge)(k)),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1173]

[End]

[iTC\_CC\_ATP-SwRS-0622]

CBIvariantReportLastAge，记录最新收到的联锁消息已存活的时间。

def [CBIvariantReportLastAge](#CBIvariantReportLastAge)(cbi, k):

return Message.[LastAge](#LastAge)([CBIvariantReportAvailable](#CBIvariantReportAvailable)(cbi, k),

[CBIvariantReport](#CBIvariantReport)(cbi).CcLoopHour,

[CBIvariantReportLastAge](#CBIvariantReportLastAge)(cbi, k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1173]

[End]

[iTC\_CC\_ATP-SwRS-0623]

BMcbiVariants, 当来自CBI的变量可用时，存储CBI变量；其他时候保持不变。

def [BMcbiVariants](#BMcbiVariants)(cbi, k):

if ([CBIvariantReportAvailable](#CBIvariantReportAvailable)(cbi, k)):

for idx in [range](#range)(0, [CBIvariantReport](#CBIvariantReport).NumberOrVariants):

[BMcbiVariants](#BMcbiVariants)[cbi].Variants[idx] = [CBIvariantReport](#CBIvariantReport).Variant[idx]

else:

pass

return [BMcbiVariants](#BMcbiVariants)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1179]

[End]

**NOTES:**

考虑到ATP程序镜像大小和代码执行效率，在设计时最多存储并维护3个CBI的变量消息（由项目设计保证，列车最多跨2个联锁区段）。ATP对于新收到并解析完成的联锁消息存储规则如下：

* 如果之前已经存储有该联锁消息，则使用新消息将其覆盖；
* 否则，如果尚有空的存储空间，则将新消息存储在空的位置；
* 否则，使用新消息覆盖掉既有的3个消息中最旧的联锁消息。

[iTC\_CC\_ATP-SwRS-0624]

CBIvariantAge，CBI变量的有效存活时间，最大值为**REPORT\_AGE\_MAX**。

* 该值与CBIvariantReportLastAge的区别是在判断回复远端ATP消息时，使用[OtherATPmaxTime](#OtherATPmaxTime)进行计算，在判断有效期时导向安全侧。

def [CBIvariantAge](#CBIvariantAge)(cbi, k):

if (Initialization

or [CBIvariantAge](#CBIvariantAge)(k-1) >= **REPORT\_AGE\_MAX**):

return **REPORT\_AGE\_MAX**

elif ([CBIvariantReportAvailable](#CBIvariantReportAvailable)(cbi, k)

and Message.[ReplyLocalCC](#ReplyLocalCC)([CBIvariantReport](#CBIvariantReport)(cbi).CcLoopHour)):

return (1 + Message.[ModularSub](#ModularSub)([ATPtime](#ATPtime)(k), [CBIvariantReport](#CBIvariantReport)(cbi).CcLoopHour))

elif ([CBIvariantReportAvailable](#CBIvariantReportAvailable)(cbi, k)

and Message.[ReplyDistantCC](#ReplyDistantCC)([CBIvariantReport](#CBIvariantReport)(cbi).CcLoopHour)):

return (1 + Message.[ModularSub](#ModularSub)([OtherATPmaxTime](#OtherATPmaxTime)(k), [CBIvariantReport](#CBIvariantReport)(cbi).CcLoopHour))

else:

return (1 + [CBIvariantAge](#CBIvariantAge)(cbi, k-1))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1179]

[End]

[iTC\_CC\_ATP-SwRS-0625]

CBIvariantLowValidity，判断是否在CBI无线的短有效期内，用于PZ的监控。 在CBTC或者使用来自BM信标变量的情况下，该值为真。

def [CBIvariantLowValidity](#CBIvariantLowValidity)(cbi, k):

if (not [BlockModeUsed](#BlockModeUsed)(k)

or not [CBIvariantMoreAvailableThanBeacon](#CBIvariantMoreAvailableThanBeacon)(cbi, k)

or [CBIvariantAge](#CBIvariantAge)(cbi, k) <= [ATPsetting](#ATPsetting).VariantsBMlowValidityTime):

return **True**

else:

return **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1179], [iTC\_CC-SyAD-1181], [iTC\_CC\_ATP\_SwHA-0236]

[End]

[iTC\_CC\_ATP-SwRS-0626]

BMcbiVariantValue，根据联锁变量索引，获得CBI的变量。

def [BMcbiVariantValue](#BMcbiVariantValue)(CbiId, VarIndex, k):

if ([CBIvariantAge](#CBIvariantAge)(CbiId, k) > [ATPsetting](#ATPsetting).VariantsBMfullValidityTime):

return **False**

else:

return [BMcbiVariants](#BMcbiVariants)[CbiId].Variants[VarIndex]

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1179], [iTC\_CC-SyAD-1183], [iTC\_CC\_ATP\_SwHA-0235]

[End]

[iTC\_CC\_ATP-SwRS-0754]

AppliedCBIvariantLoopHour，记录当前使用的CBI的变量的CC时间，供CCNV使用。

def [AppliedCBIvariantLoopHour](#AppliedCBIvariantLoopHour)(cbiId, k):

if ([CBIvariantReportAvailable](#CBIvariantReportAvailable)(cbiId, k)):

return [CBIvariantReport](#CBIvariantReport)(cbiId).CcLoopHour

else:

return [AppliedCBIvariantLoopHour](#AppliedCBIvariantLoopHour)(cbiId, k-1)

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0408], [iTC\_CC-SyAD-1179]

[End]

#### BM variants outputs

在BM模式下，在每周期应明确使用来自CBI无线的变量还是BM信标的变量。使用这两种变量均是安全的，需要从可用性角度，判断哪种更加新，则使用该变量。原则如下：

* 初始化时，或项目未配置CBI无线，则使用BM信标变量；
* 否则，如果列车失位或[LocationPathKnown](#LocationPathKnown)未知，则使用BM信标变量；
* 否则，如果之前未收到BM信标，或BM信标变量不可用，则使用CBI无线变量；
* 否则，如果CBI无线变量在有效期内；且比之前收到的BM信标加传输延迟更新；且该无线变量是在BMCP点上游的接收窗之后收到的，则使用CBI无线变量。
* 否则，使用BM信标变量。

[iTC\_CC\_ATP-SwRS-0627]

CBIvariantMoreAvailableThanBeacon，通过比较最后一次收到的BM信标的有效期，和对应变量所在该联锁区的无线变量，判断对于该变量，是使用来自CI无线的变量而非来自信标的变量。

ATP shall use the more recent message from beacons and CBI radio.

def [CBIvariantMoreAvailableThanBeacon](#CBIvariantMoreAvailableThanBeacon)(CbiId, k):

if (Initialization

or not [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k-1)

or not [ATPsetting](#ATPsetting).BlockModeThroughRadio(k)):

return **False**

else:

return ([UsedBMbeaconId](#UsedBMbeaconId)(k) is **None**

or ([CBIvariantAge](#CBIvariantAge)(CbiId, k) <= [ATPsetting](#ATPsetting).VariantsBMfullValidityTime

and ([CBIvariantReportLastAge](#CBIvariantReportLastAge)(CbiId, k)

<= [BMbeaconReadAge](#BMbeaconReadAge)(k) + [ATPsetting](#ATPsetting).VariantsBMradioPriorityDelay)

and ([CBIvariantReportLastAge](#CBIvariantReportLastAge)(CbiId, k) <= [CBIminProductionAge](#CBIminProductionAge)(CbiId, k))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0108], [iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-1163], [iTC\_CC\_ATP\_SwHA-0275]

[End]

**NOTES：**

判断条件[CBIvariantReportLastAge](#CBIvariantReportLastAge) <= [CBIminProductionAge](#CBIminProductionAge)，表明当前使用的无线消息是列车进入BMCP点上游 “Reception Windows”之后收到的新的消息（或者列车还未经过BMCP点），因此可以使用。

* 若不满足这个条件，表明列车经过了BMCP点，但未在“Reception Windows”内收到新的无线消息，因此不能相信；
* 此时应使用来自信标的消息（该信标应当布置在Reception Windows之中，且由配置Vital zone保证不能丢失）。

可参考需求[CBIminProductionAgeSinceSSAcrossing](#CBIminProductionAgeSinceSSAcrossing)和[CBIminProductionAge](#CBIminProductionAge)。

[iTC\_CC\_ATP-SwRS-0628]

BMvariantValue，统一来自BM信标和CBI无线的BM变量

def [BMvariantValue](#BMvariantValue)(Variant, k):

if ([CBIvariantMoreAvailableThanBeacon](#CBIvariantMoreAvailableThanBeacon)(Variant.Cbi.Id, k)):

return [BMcbivariantValue](#BMcbivariantValue)(Variant.Cbi.Id, Variant.Cbi.Index, k)

else:

return [BMbeaconVariantValue](#BMbeaconVariantValue)(Variant.LineSec.Id, Variant.LineSec.Index, k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-0296], [iTC\_CC-SyAD-1179] , [iTC\_CC\_ATP\_SwHA-0275]

[End]

[iTC\_CC\_ATP-SwRS-0629]

BMvariantRemainingTime，BM变量的剩余有效期

def [BMvariantRemainingTime](#BMvariantRemainingTime)(cbi, k):

if (not [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)(k)):

return 0

elif ([CBIvariantMoreAvailableThanBeacon](#CBIvariantMoreAvailableThanBeacon)(cbi, k)):

return [max](#max)(0, [ATPsetting](#ATPsetting).VariantsBMfullValidityTime - [CBIvariantAge](#CBIvariantAge)(cbi, k))

else:

return [max](#max)(0, [ATPsetting](#ATPsetting).VariantsBMfullValidityTime - [BMbeaconReadAge](#BMbeaconReadAge)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0290], [iTC\_CC-SyAD-1189], [iTC\_CC\_ATP\_SwHA-0235]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [AppliedCBIvariantLoopHour](#AppliedCBIvariantLoopHour) | √ | √ | NUMERIC\_32 |
| [BMbeaconReadAge](#BMbeaconReadAge) | √ | √ | NUMERIC\_32 |
| [BMvariantRemainingTime](#BMvariantRemainingTime) | √ | √ | NUMERIC\_32 |
| [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid) | √ | √ | BOOLEAN |
| [BMvariantValue](#BMvariantValue) | √ | × | BOOLEAN |
| [BlockModeUsed](#BlockModeUsed) | √ | √ | BOOLEAN |
| [CBIvariantMoreAvailableThanBeacon](#CBIvariantMoreAvailableThanBeacon) | √ | √ | BOOLEAN |
| [CBIvariantReportLastAge](#CBIvariantReportLastAge) | √ | √ | NUMERIC\_32 |

## F15-Manage Variants in CBTC Mode

管理CBTC模式下的变量

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [ATPtime](#ATPtime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid) | Internal | F14-Manage Variants in Block Mode |
| [BMvariantValue](#BMvariantValue) | Internal | F14-Manage Variants in Block Mode |
| [EOA\_Report](#EOA_Report) | External | Interface with ZC |
| [OtherATP](#OtherATP) | Internal | F13-Manage Loop Hour with Distant ATP |
| [OtherATPmaxTime](#OtherATPmaxTime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [OtherATPmessagevalid](#OtherATPmessagevalid) | Internal | F13-Manage Loop Hour with Distant ATP |
| [OtherATPminTime](#OtherATPminTime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [SafeApplicationVersion](#SafeApplicationVersion) | Internal | F11-Acquire Configuration Data |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [VariantReport](#VariantReport) | External | Interface with ZC |
| [VersionAuthorization](#VersionAuthorization) | External | Interface with LC |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [CBTCvariantValue](#CBTCvariantValue) | √ | × | BOOLEAN |
| [EOAReportReceived](#EOAReportReceived) | √ | √ | BOOLEAN |
| [EOAgroundAge](#EOAgroundAge) | √ | √ | NUMERIC\_32 |
| [LastEOAReportAge](#LastEOAReportAge) | √ | √ | NUMERIC\_32 |
| [LastVersionReportAge](#LastVersionReportAge) | √ | √ | NUMERIC\_32 |
| [ReceivedEOAreport](#ReceivedEOAreport) | √ | √ | ST\_EOA\_RCV |
| [ReceivedVariantReport](#ReceivedVariantReport) | √ | × | ST\_VARIANT\_RCV |
| [ReceivedVersionMessages](#ReceivedVersionMessages) | √ | √ | ST\_VERSION\_RCV |
| [SameVersionWithDistantCore](#SameVersionWithDistantCore) | √ | √ | BOOLEAN |
| [VariantGroundAge](#VariantGroundAge) | √ | √ | NUMERIC\_32 |
| [VariantReportReceived](#VariantReportReceived) | √ | √ | BOOLEAN |
| [VersiionAuthorizationAvailable](#VersiionAuthorizationAvailable) | √ | √ | BOOLEAN |
| [VersiionAuthorizationReceived](#VersiionAuthorizationReceived) | √ | √ | BOOLEAN |
| [ZCmessageReady](#ZCmessageReady) | √ | √ | BOOLEAN |

### Processing

#### Version authorization by LC

[iTC\_CC\_ATP-SwRS-0630]

VersionAuthorizationReceived，收到版本授权

def [VersionAuthorizationReceived](#VersionAuthorizationReceived)(lcId, k):

return [Message.Received](#Received)([VersionAuthorization](#VersionAuthorization)(lcId), k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0378]

[End]

[iTC\_CC\_ATP-SwRS-0103]

VersionAuthorizationAvailable，LC版本授权消息可用

def VersionAuthorizationAvailable(lcId, k):

return Message.[Available](#Available)([VersionAuthorizationReceived](#VersionAuthorizationReceived)(lcId, k),

[VersionAuthorization](#VersionAuthorization)(lcId).CcLoopHour,

[ATPsetting](#ATPsetting).VersionsValidityTime,

[LastVersionReportAge](#LastVersionReportAge)(lcId, k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0378], [iTC\_CC\_ATP\_SwHA-0021], [iTC\_CC\_ATP\_SwHA-0019]

[End]

[iTC\_CC\_ATP-SwRS-0453]

LastVersionReportAge，记录从上次收到LC的版本信息到现在的时间。

def [LastVersionReportAge](#LastVersionReportAge)(lcId, k):

return Message.[LastAge](#LastAge)([VersionAuthorizationAvailable](#VersionAuthorizationAvailable)(lcId, k),

[VersionAuthorization](#VersionAuthorization)(lcId).CcLoopHour,

[LastVersionReportAge](#LastVersionReportAge)(lcId, k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0378]

[End]

[iTC\_CC\_ATP-SwRS-0104]

ReceivedVersionMessages，用于存储从LC收到的**MAX\_ZC\_NB**个ZC区的授权信息。由于每个ZC分属不同的LC管理，因此收到特定的LC消息时应仅更新其所对应ZC的版本授权状态。

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_VERSION\_RCV | |  |  |
|  | ValidityTime | NUMERIC\_32 | 版本授权信息过期时间 |
|  | VitalAuthorization[**MAX\_ZC\_NB**] | BOOLEAN | ZC区授权信息 |

def [ReceivedVersionMessages](#ReceivedVersionMessages)(LcId, k):

if (Initialization):

ReceivedVersionMessages = **None**

elif ([VersionAuthorizationAvailable](#VersionAuthorizationAvailable)(LcId, k)):

if (Message.[ReplyLocalCC](#ReplyLocalCC)([VersionAuthorization](#VersionAuthorization)(LcId).CcLoopHour)):

NewValidity = ([VersionAuthorization](#VersionAuthorization)(LcId).CcLoopHour

+ [ATPsetting](#ATPsetting).VersionsValidityTime)

else:

NewValidity

= ([ATPtime](#ATPtime)(k) + [ATPsetting](#ATPsetting).VersionsValidityTime

- ([OtherATPmaxTime](#OtherATPmaxTime)(k) - [VersionAuthorization](#VersionAuthorization)(LcId).CcLoopHour))

for ZcId in [range](#range)(0, **MAX\_ZC\_NB**):

if ([TrackMap](#TrackMap).Zc[ZcId].LcId == LcId):

[ReceivedVersionMessages](#ReceivedVersionMessages)[ZcId].VitalAuthorization

= [VersionAuthorization](#VersionAuthorization)(LcId).VitalAuthorization[ZcId]

[ReceivedVersionMessages](#ReceivedVersionMessages)[ZcId].ValidityTime = NewValidity

else:

pass

else:

[ReceivedVersionMessages](#ReceivedVersionMessages) = [ReceivedVersionMessages](#ReceivedVersionMessages)(k-1)

return [ReceivedVersionMessages](#ReceivedVersionMessages)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0220], [iTC\_CC-SyAD-0373], [iTC\_CC-SyAD-0379], [iTC\_CC-SyAD-0380], [iTC\_CC\_ATP\_SwHA-0023], [iTC\_CC\_ATP\_SwHA-0177]

[End]

[iTC\_CC\_ATP-SwRS-0631]

VersionAuthorizedByLC，获取ZC的版本授权状态

def [VersionAuthorizedByLC](#VersionAuthorizedByLC)(ZcId, k):

if (Message.IsMoreRecent

([ReceivedVersionMessages](#ReceivedVersionMessages)([TrackMap](#TrackMap).Zc[ZcId].LcId ,k)[zcId].ValidityTime,

[ATPtime](#ATPtime)(k))):

return [ReceivedVersionMessages](#ReceivedVersionMessages)([TrackMap](#TrackMap).Zc[ZcId].LcId, k)[zcId].VitalAuthorization

else:

return **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0300]

[End]

#### ZC message availability

[iTC\_CC\_ATP-SwRS-0093]

SameVersionWithDistantCore，比较来自远端ATP的安全软件，项目配置数据，以及线路地图版本号与本端是否一致

The local ATP shall compare the information from the redundant ATP to ensure the consistency, which includes versions of vital software, project configuration data and the track map.

def [SameVersionWithDistantCore](#SameVersionWithDistantCore)(k):

if ([OtherATPmessageValid](#OtherATPmessageValid)(k)

and ([OtherATP](#OtherATP).SafetyParameterVersion == [ATPsetting](#ATPsetting).SafetyParameterVersion)

and ([OtherATP](#OtherATP).SafetyApplicationVersion == [SafeApplicationVersion](#SafeApplicationVersion))):

for ZcId in [range](#range)(0, **MAX\_ZC\_NB**):

if ([OtherATP](#OtherATP).ZcVersion[ZcId] != [TrackMap](#TrackMap).[ZC](#ZC)[ZcId].Version):

return **False**

else:

continue

else:

return **True**

else:

return **False**

#Category= Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source= [iTC\_CC-SyAD-0965]

[End]

[iTC\_CC\_ATP-SwRS-0632]

EOAReportReceived，收到EOA消息

def [EOAReportReceived](#EOAReportReceived)(k):

return [Message.Received](#Received)([EOAReport](#EOAReport), k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0153]

[End]

[iTC\_CC\_ATP-SwRS-0105]

ZCmessageReady，表示本周期收到了有效的来自ZC的EOA和变量消息。

* 当前时间大于消息中的ccLoopHour
* 消息中的ccLoopHour+EOA有效期，应大于当前时间

[ZCmessageReady](#ZCmessageReady) represents that an available EOA and variants message from [ZC](#ZC) received in this cycle.

def [ZCmessageReady](#ZCmessageReady)(k):

return (Message.[Available](#Available)([EOAReportReceived](#EOAReportReceived)(k),

[EOA\_Report](#EOA_Report).CcLoopHour,

[ATPsetting](#ATPsetting).EOAvalidityTime,

[LastEOAReportAge](#LastEOAReportAge)(k-1),

k)

and ([VersionAuthorizedByLC](#VersionAuthorizedByLC)([SSIDofZC](#SSIDofZC), k))

and (Message.[ReplyLocalCC](#ReplyLocalCC)([EOA\_Report](#EOA_Report).CcLoopHour)

or [SameVersionWithDistantCore](#SameVersionWithDistantCore)(k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0153], [iTC\_CC-SyAD-0155], [iTC\_CC-SyAD-0156], [iTC\_CC-SyAD-0158], [iTC\_CC-SyAD-0962], [iTC\_CC-SyAD-0965], [iTC\_CC-SyAD-0966], [iTC\_CC\_ATP\_SwHA-0026], [iTC\_CC\_ATP\_SwHA-0251]

[End]

[iTC\_CC\_ATP-SwRS-0108]

LastEOAReportAge，数值型，上次发出loc-report的周期数减去EOA在ZC端消耗的时间（CC周期数）。

[LastEOAReportAge](#LastEOAReportAge) represents the value calculated by current ATP time minus the previous loc-report number and the EOA consuming time in [ZC](#ZC).

def [LastEOAReportAge](#LastEOAReportAge)(k):

return Message.[LastAge](#LastAge)([ZCmessageReady](#ZCmessageReady)(k),

[EOA\_Report](#EOA_Report).CcLoopHour,

[LastEOAReportAge](#LastEOAReportAge)(k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0155], [iTC\_CC-SyAD-0965]

[End]

[iTC\_CC\_ATP-SwRS-0106]

EOAgroundAge，数值型，在收到EOA消息时，其时间已经消耗了几个CC的周期。需同时维护WithoutSpaceEoa和普通EOA。

[EOAgroundAge](#EOAgroundAge) stands for the number of CC cycle when receiving the EOA information.

def [EOAgroundAge](#EOAgroundAge)(k):

if (Initialization):

[EOAgroundAge](#EOAgroundAge).WithoutSpacing = **REPORT\_AGE\_MAX**

[EOAgroundAge](#EOAgroundAge).Classic = **REPORT\_AGE\_MAX**

elif ([ZCmessageReady](#ZCmessageReady)(k)):

[EOAgroundAge](#EOAgroundAge).WithoutSpacing = ([round.ceil](#roundceil)

(([EOA\_Report](#EOA_Report).MessageContainerCreationTime

- [EOA\_Report](#EOA_Report).WithoutSpacingEoaCreationTime)

\* SYNCHRODATE\_TIME\_UNIT\_MS / ATP\_CYCLE\_TIME\_MS))

[EOAgroundAge](#EOAgroundAge).Classic = [round.ceil](#roundceil)(([EOA\_Report](#EOA_Report).MessageContainerCreationTime

- [EOA\_Report](#EOA_Report).EoaCreationTime)

\* SYNCHRODATE\_TIME\_UNIT\_MS / ATP\_CYCLE\_TIME\_MS)

else:

[EOAgroundAge](#EOAgroundAge) = [EOAgroundAge](#EOAgroundAge)(k-1)

return [EOAgroundAge](#EOAgroundAge)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0156], [iTC\_CC\_ATP\_SwHA-0179]

[End]

[iTC\_CC\_ATP-SwRS-0107]

ReceivedEOAreport，判断当新收到EOA消息的有效期大于之前存储EOA消息有效期时，更新EOA。需同时维护WithoutSpaceEoa和普通EOA。当存储的EOA消息过期后，清除该消息。

|  |  |  |
| --- | --- | --- |
| Identification | | Description |
| ST\_EOA\_RCV | |  |
|  | .TrainFrontEnd | EOA对应的车头 |
|  | .Classic.ValidityTime | 普通EOA的有效期截止时间 |
|  | .Classic.Type | 普通EOA的类型 |
|  | .Classic.Location | 普通EOA的位置 |
|  | .WithoutSpacing.ValidityTime | 在SMI区域使用的EOA的有效期截止时间 |
|  | .WithoutSpacing.Type | SMI区使用的EOA的类型 |
|  | .WithoutSpacing.Location | SMI区使用的EOA的位置 |
|  | .CcLoopHour | EOA消息回复的CC时间 |

def [ReceivedEOAreport](#ReceivedEOAreport)(k):

if (Initialization):

[ReceivedEOAreport](#ReceivedEOAreport) = **None**

elif ([ZCmessageReady](#ZCmessageReady)(k)):

[ReceivedEOAreport](#ReceivedEOAreport).[TrainFrontEnd](#TrainFrontEnd) = [EOA\_Report](#EOA_Report).[TrainFrontEnd](#TrainFrontEnd)

[ReceivedEOAreport](#ReceivedEOAreport).Classic = [UpdateReceivedEoa](#UpdateReceivedEoa)([EOA\_Report](#EOA_Report).CcLoopHour,

[EOAgroundAge](#EOAgroundAge)(k).Classic,

[EOA\_Report](#EOA_Report).Classic,

[ReceivedEOAreport](#ReceivedEOAreport)(k-1).Classic)

[ReceivedEOAreport](#ReceivedEOAreport).WithoutSpacing = ([UpdateReceivedEoa](#UpdateReceivedEoa)

([EOA\_Report](#EOA_Report).CcLoopHour,

[EOAgroundAge](#EOAgroundAge)(k).WithoutSpacing,

[EOA\_Report](#EOA_Report).WithoutSpacing,

[ReceivedEOAreport](#ReceivedEOAreport)(k-1).WithoutSpacing))

[ReceivedEOAreport](#ReceivedEOAreport).CcLoopHour = [EOA\_Report](#EOA_Report).CcLoopHour

else:

[ReceivedEOAreport](#ReceivedEOAreport) = [ReceivedEOAreport](#ReceivedEOAreport)(k-1)

if ([Message.IsMoreRecent](#IsMoreRecent)([ATPtime](#ATPtime)(k), [ReceivedEOAreport](#ReceivedEOAreport).Classic.ValidityTime)):

clean\_reseived\_eoa\_classic

if ([Message.IsMoreRecent](#IsMoreRecent)([ATPtime](#ATPtime)(k), [ReceivedEOAreport](#ReceivedEOAreport).WithoutSpacing.ValidityTime)):

clean\_reseived\_eoa\_without\_space

return [ReceivedEOAreport](#ReceivedEOAreport)

其中UpdateReceivedEoa定义如下:

def [UpdateReceivedEoa](#UpdateReceivedEoa)(NewEoaLoopHour, EoaGroundAge, NewReceivedEoa, PreviousReceivedEoa):

if ([Message.ReplyLocalCc](#ReplyLocalCc)(NewEoaLoopHour)):

NewValidity = (NewEoaLoopHour - EoaGroundAge + [ATPsetting](#ATPsetting).EOAvalidityTime)

else:

NewValidity = ([ATPtime](#ATPtime)(k) - EoaGroundAge + [ATPsetting](#ATPsetting).EOAvalidityTime

- ([OtherATPmaxTime](#OtherATPmaxTime)(k) - NewEoaLoopHour))

if ([Message.IsMoreRecent](#IsMoreRecent)(NewValidity, [ATPtime](#ATPtime)(k))

and ([Message.IsMoreRecent](#IsMoreRecent)(NewValidity, PreviousReceivedEoa.ValidityTime))):

return NewReceivedEoa

else:

return PreviousReceivedEoa

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0153], [iTC\_CC-SyAD-0156], [iTC\_CC-SyAD-0158], [iTC\_CC\_ATP\_SwHA-0025], [iTC\_CC\_ATP\_SwHA-0028], [iTC\_CC-SyAD-1005]

[End]

#### Variants in CBTC mode

[iTC\_CC\_ATP-SwRS-0109]

VariantGroundAge，将ZC端的变量生存时间转换为CC周期数

[VariantGroundAge](#VariantGroundAge) shows the survival time of the variants in [ZC](#ZC).

def [VariantGroundAge](#VariantGroundAge)(lineSec, k):

[VariantGroundAge](#VariantGroundAge) = [round.ceil](#roundceil)(([EOA\_Report](#EOA_Report).MessageContainerCreationTime

- [VariantReport](#VariantReport)(lineSec).CreationTime)

\* SYNCHRODATE\_TIME\_UNIT\_MS / ATP\_CYCLE\_TIME\_MS)

return [VariantGroundAge](#VariantGroundAge)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0160]

[End]

[iTC\_CC\_ATP-SwRS-0633]

VariantReportReceived，收到ZC变量消息

def [VariantReportReceived](#VariantReportReceived)(LineSec, k):

return [Message.Received](#Received)([VariantReport](#VariantReport)(LineSec), k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0159]

[End]

[iTC\_CC\_ATP-SwRS-0110]

ReceivedVariantReport，存储来自ZC的变量消息，如ST\_VARIANT\_RCV所示，按照LineSection进行存储：

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_VARIANT\_RCV | |  |  |
|  | ValidityTime | NUMERIC\_32 | 变量过期时间 |
|  | Variants | ST\_VARIANT\_REPORT | 该LineSection的变量 |

def [ReceivedVariantReport](#ReceivedVariantReport)(LineSec, k):

if ([ZCmessageReady](#ZCmessageReady)(k)

and [VariantReportReceived](#VariantReportReceived)(LineSec, k)):

if (Message.ReplyLocalCc([ReceivedEOAreport](#ReceivedEOAreport)(k).CcLoopHour)):

NewValidity = ([ReceivedEOAreport](#ReceivedEOAreport)(k).CcLoopHour

- [VariantGroundAge](#VariantGroundAge)(LineSec, k)

+ [ATPsetting](#ATPsetting).VariantsCBTCvalidityTime)

else:

NewValidity = ([ATPtime](#ATPtime)(k) - [VariantGroundAge](#VariantGroundAge)(LineSec, k)

+ [ATPsetting](#ATPsetting).VariantsCBTCvalidityTime

- ([OtherATPmaxTime](#OtherATPmaxTime)(k) - [ReceivedEOAreport](#ReceivedEOAreport)(k).CcLoopHour))

if (Message.IsMoreRecent(NewValidity, [ATPtime](#ATPtime)(k))

and ([Message.IsMoreRecent](#IsMoreRecent)

(NewValidity, [ReceivedVariantReport](#ReceivedVariantReport)[LineSec](k-1).ValidityTime))):

[ReceivedVariantReport](#ReceivedVariantReport)[LineSec].ValidityTime = NewValidity

[ReceivedVariantReport](#ReceivedVariantReport)[LineSec].Variants = [VariantReport](#VariantReport)(LineSec, Variants)

else:

[ReceivedVariantReport](#ReceivedVariantReport)[LineSec] = [ReceivedVariantReport](#ReceivedVariantReport)[LineSec](k-1)

else:

[ReceivedVariantReport](#ReceivedVariantReport)[LineSec] = [ReceivedVariantReport](#ReceivedVariantReport)[LineSec](k-1)

return [ReceivedVariantReport](#ReceivedVariantReport)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0159], [iTC\_CC-SyAD-0160], [iTC\_CC-SyAD-0161], [iTC\_CC\_ATP\_SwHA-0030], [iTC\_CC\_ATP\_SwHA-0031], [iTC\_CC-SyAD-0153]

[End]

**NOTES：**

在CBTC运营模式下，ATP通过解析来自ZC的消息来获取线路上的变量状态。ZC所发送的变量消息以LineSection为单位，可能发送当前ZC区域以及相邻下个ZC区域的LineSection的变量。ATP对于变量的存储和校验以及时间有效性维护，也应以LineSection为单位。

In the CBTC mode, ATP gets the variants by parse the [ZC](#ZC) message. The variants sending from [ZC](#ZC) are categorized by line section. Moreover, it will send the line section variants in the current [ZC](#ZC) area and the adjacent one. ATP will record and check the variants and maintains the time effectiveness.

[iTC\_CC\_ATP-SwRS-0150]

CBTCvariantValue，维护CBTC下变量的值

* 如果变量有效期大于当前时间，则使用该变量；否则为限制状态

ATP shall maintain the validation of CBTC variants message from ZC. if the validation timeout, ATP should set all CBTC variants to restrictive state.

def [CBTCvariantValue](#CBTCvariantValue)(Variant, k):

if ([ReceivedVariantReport](#ReceivedVariantReport)(Variant.LineSec.Id, k).ValidityTime > [ATPtime](#ATPtime)(k)):

return [ReceivedVariantReport](#ReceivedVariantReport)(Variant.LineSec.Id, k).Status(Variant.LineSec.Index)

else:

return **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0162], [iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0299], [iTC\_CC-SyAD-0159], [iTC\_CC\_ATP\_SwHA-0031]

[End]

[iTC\_CC\_ATP-SwRS-0634]

VariantValue，统一CBTC和BM下的变量

def [VariantValue](#VariantValue)(Variant, k):

if ([BlockModeUsed](#BlockModeUsed)(k)):

return [BMvariantValue](#BMvariantValue)(Variant, k)

else:

return [CBTCvariantValue](#CBTCvariantValue)(Variant, k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-0296], [iTC\_CC-SyAD-0299]

[End]

### Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Safety Critical | | | Observable | | Logical Type |
| [VariantValue](#VariantValue) | | √ | × | | BOOLEAN | |
| [VersiionAuthorizedByLC](#VersiionAuthorizedByLC) | | √ | √ | | BOOLEAN | |

## F2-Measure Train Kinematics

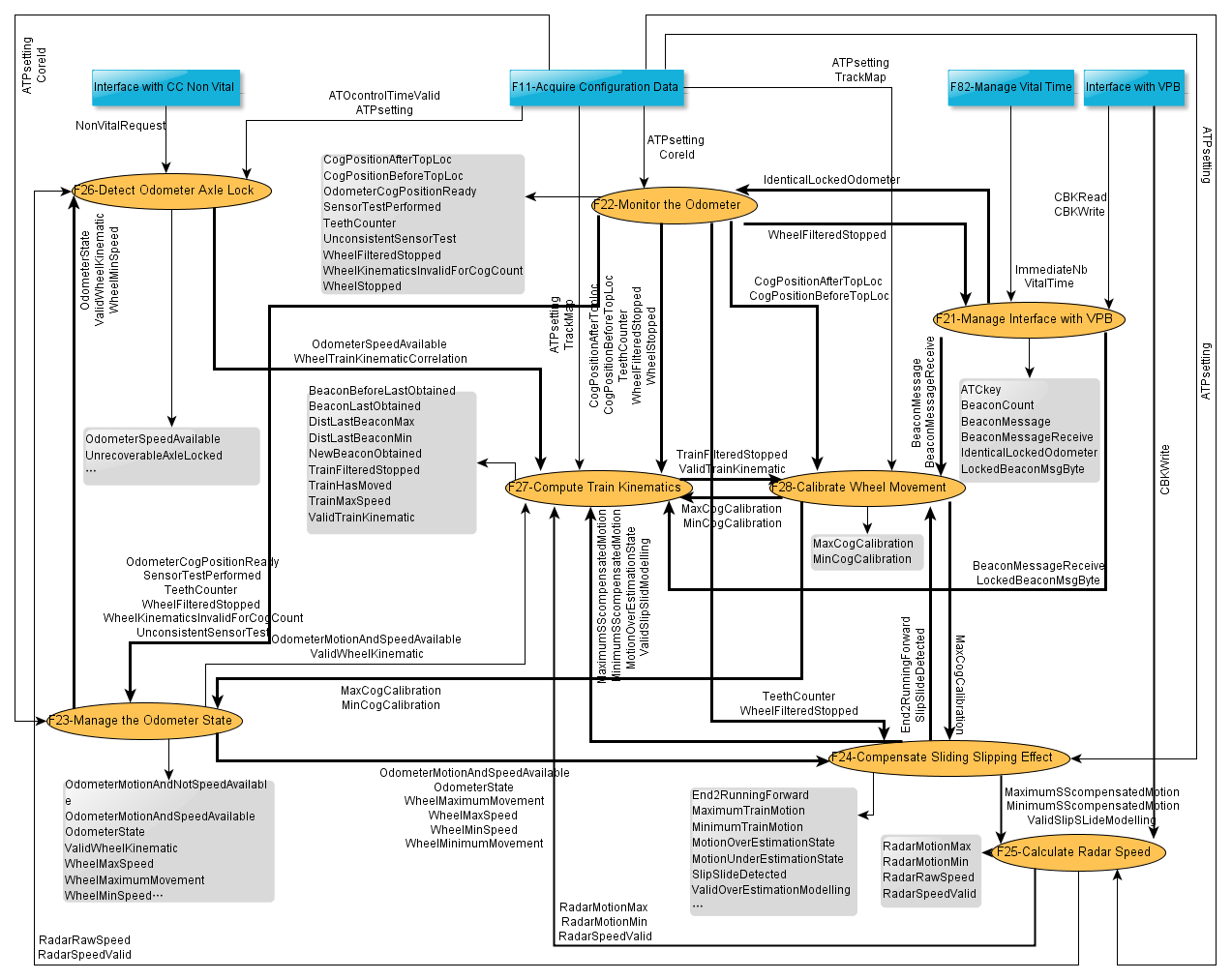


Figure 5‑6 SART modeling of function F2

## F21-Manage Interface with VPB

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [CBKRead](#CBKRead) | External | Interface with VPB |
| [CBKWrite](#CBKWrite) | External | Interface with VPB |
| [ImmediateNb](#ImmediateNb) | Internal | F82-Manage Vital Time |
| [VitalTime](#VitalTime) | Internal | F82-Manage Vital Time |
| [WheelFilteredStopped](#WheelFilteredStopped) | Internal | F22-Monitor the Odometer |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [BeaconChecksumFailure](#BeaconChecksumFailure) | √ | √ |  |
| [BeaconMsgByte](#BeaconMsgByte) | √ | × | NUMERIC\_32 |
| [BeaconMsgReady](#BeaconMsgReady) | √ | × |  |
| [LockedBeaconMsgReady](#LockedBeaconMsgReady) | √ | √ |  |
| [LockedOdometer](#LockedOdometer) | √ | √ | ST\_ODOMETER\_IMM |
| [LockedTopLocCounter](#LockedTopLocCounter) | √ | √ | NUMERIC\_32 |
| [OdometerImm](#OdometerImm) | √ | × | ST\_ODOMETER\_IMM |
| [SensorTestFlag](#SensorTestFlag) | √ | √ |  |
| [TopLocCounter](#TopLocCounter) | √ | √ | NUMERIC\_32 |

### Processing

#### Odometer information

编码里程计运行过程中的安全性由其齿数齿号一致性的检查来保证，因此ATP需在每次中断中锁存VPB齿数和齿号寄存器中的值，如Table 5‑6所示，在主任务中进行判断。而当编码里程计未发生转动时，ATP还需驱动VPB对里程计的各路传感器进行测试，并通过传感器返回的导通状态来判断里程计是否工作正常。

The consistency of cog count and cog code makes sure the safety during the processing of coded odometer, so ATP needs to save the value of cog count and cog code in each interrupt as shown in Table 5‑6, and judge it in the main task. However when the coded odometer does not move, ATP needs to driver VPB board in order to test every sensor in the odometer. In addition, ATP needs to judge the working status of odometer through the conduction status returning from the sensor.

Table 5‑6 Odometer information in interrupt

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_ODOMETER\_IMM | |  |  |
|  | CogCounter | NUMERIC\_32 | 该中断齿数 |
|  | TopLocValid | NUMERIC\_32 | 该中断检测到Top-loc |
|  | CogCode | NUMERIC\_32 | 该中断齿号寄存器值 |
|  | CalCogCounter | NUMERIC\_32 | 当top-loc发生时VPB锁存的齿数 |
|  | SensorTesting | BOOLEAN | 该中断是否进行传感器测试 |
|  | TestResult | ENUM\_SENSOR\_TEST\_RESULT | 该中断传感器测试结果 |
|  | A1 | ENUM\_SENSOR\_STATUS | 该中断中传感器1的状态 |
|  | A2 | ENUM\_SENSOR\_STATUS | 该中断中传感器2的状态 |
|  | A3 | ENUM\_SENSOR\_STATUS | 该中断中传感器3的状态 |

[iTC\_CC\_ATP-SwRS-0113]

使用OdometerImm记录每次中断中的VPBWrite寄存器编码里程计相关属性变化情况，其结构如Table 5‑6所示。

[OdometerImm](#OdometerImm) records the changes of VPBWrite register in each interrupt, structured as Table 5‑6.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0130], [iTC\_CC\_VLE-2-VPB-2-SyID-0023], [iTC\_CC\_VLE-2-VPB-2-SyID-0024]

[End]

[iTC\_CC\_ATP-SwRS-0114]

中断中，当VPB板检测到Top-loc信号后，ATP累加本周期的TopLocCounter；ATP只有当检测到同一个中断中VPB寄存器的Top-loc和BMR信号同时为**True**时，才设置本中断的[OdometerImm](#OdometerImm).[TopLocValid](#TopLocValid)和本周期的BeaconMsgReady为**True**，并将信标寄存器中的数据存储到BeaconMsgByte[**MAX\_BEACON\_DATA\_SIZE**]中。

* 其中，由于上下模块同步算法，可能出现两个第0中断的情况，此时如果第一个第0中断中读到了VPB的[TopLocValid](#TopLocValid)，则第二个第0中断不覆盖该Top-loc信息。
* 在每周期开始，清除BeaconMsgReady和TopLocCounter。

In the interrupt, when the [TopLocValid](#TopLocValid) detected by VPB board regarded as **True**, [OdometerImm](#OdometerImm).[TopLocValid](#TopLocValid) and [BeaconMsgReady](#BeaconMsgReady) set as **True**. Adding [TopLocCounter](#TopLocCounter), the data from beacon savor is saved into [BeaconMsgByte](#BeaconMsgByte)[**MAX\_BEACON\_DATA\_SIZE**].

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0130], [iTC\_CC-SyAD-0968], [iTC\_CC\_ATP\_SwHA-0180], [iTC\_CC\_ATP\_SwHA-0231], [iTC\_CC\_VLE-2-VPB-2-SyID-0028]

[End]

[iTC\_CC\_ATP-SwRS-0557]

SensorTestFlag，位于CPU1的ATP软件判断是否需进行传感器测试的标志。

* 判断连续**SENSOR\_TEST\_START\_TIME**时间VPB寄存器[CBKWrite](#CBKWrite).CogCounterReg未发生变化；
* 且之前SensorTestFlag为**False**；
* 且当前[WheelFilteredStopped](#WheelFilteredStopped)为**False**。

则位于VLE-2板CPU1的ATP软件，需设置SensorTestFlag为**True**，并将其发送给位于CPU2的ATP软件。

[SensorTestFlag](#SensorTestFlag) regarded as the symbol whether ATP in CPU 1 needs to check the sensor. In the interval, if the value of [CBKWrite](#CBKWrite).CogCounterReg is not changed in the continuous **SENSOR\_TEST\_START\_TIME**, and the [WheelFilteredStopped](#WheelFilteredStopped) was **False** at last cycle, the ATP of CPU1 in VLE-2 board need to set [SensorTestFlag](#SensorTestFlag) as **True** and send the data to the ATP of CPU2.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0958], [iTC\_CC\_ATP\_SwHA-0055]

[End]

[iTC\_CC\_ATP-SwRS-0558]

对于VLE-2板上CPU1的ATP软件，如果判断上周期末时的[SensorTestFlag](#SensorTestFlag)为**True**，则从本周期开始，需在每次中断中按照既定测试序列设置[OdometerImm](#OdometerImm).D1/D2/D3的值，并在相应中断中设置SensorTesting标志为**True**。

* 在一个周期的中断中，应当每间隔1个中断发送一次D1/2/3全为**POWER\_ON**；
* 其余中断中，D1/2/3可为**POWER\_ON**或**POWER\_OFF**随机值。

For the ATP of CPU1 in VLE-2 board, if it sets the [SensorTestFlag](#SensorTestFlag) of pervious end of cycle as **True**, it need to set the value of [OdometerImm](#OdometerImm).D1/D2/D3 based on the settled sequence in each interrupt, and set the SensorTesting as **True**.

* D1/2/3 shall set to **POWER\_ON** at every other interrupts;
* In other interrupt, D1/2/3 shall be set to pseudo random value.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0958], [iTC\_CC\_ATP\_SwHA-0056], [iTC\_CC\_VLE-2-VPB-2-SyID-0025]

[End]

[iTC\_CC\_ATP-SwRS-0559]

对于VLE-2板CPU-2的ATP软件，如果收到来自CPU1的[SensorTestFlag](#SensorTestFlag)为**True**时，需通过读取VPB-2板的D1/2/3寄存器，获取当前测试的D1/2/3值。

For the ATP of CPU2, if the [SensorTestFlag](#SensorTestFlag) from CPU1 is **True**, it needs to read the D1/2/3 registers of VPB-2 board and obtain the current testing value of D1/2/3.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0958], [iTC\_CC\_VLE-2-VPB-2-SyID-0026]

[End]

[iTC\_CC\_ATP-SwRS-0115]

在传感器测试过程中，对于VLE-2板的2个CPU上的ATP软件，均需检测收到的C1/2/3三路传感器测试结果与上次中断中的结果是否一致，依次判断编码里程计传感器状态[A1](#A1)，[A2](#A2)和[A3](#A3)：

During the sensor test performing, both ATP in the two CPUs of VLE-2 shall detect the consistency of the test result according to the state of C1/2/3 in the two continuous interrupt. Accordingly, ATP determines the state of three sensors: [A1](#A1), [A2](#A2), [A3](#A3).

在中断i中，用于判断单个的传感器1/2/3是处于**SENSOR\_CONDUCT**还是**SENSOR\_BLOCKED**状态，用Ai表示，条件如下：

* if D(i-1)= **POWER\_ON** & Ci=**LOW\_LEVEL**，Ai = **SENSOR\_BLOCKED**
* if D(i-1)= **POWER\_ON** & Ci=**HIGH\_LEVEL**，Ai = **SENSOR\_CONDUCT**
* if D(i-1)= **POWER\_OFF** & Ci=**LOW\_LEVEL**，保持上次测试的状态，Ai = Ai(t-1)
* if D(i-1)= **POWER\_OFF** & Ci=**HIGH\_LEVEL**，Ai = **SENSOR\_WRONG**

In the interrupt i, the rules to determine whether the state of a sensor Ai is **SENSOR\_CONDUCT** or **SENSOR\_BLOCK** are as follows:

* if D(i-1)= **POWER\_ON** & Ci=**LOW\_LEVEL**，Ai = **SENSOR\_BLOCKED**
* if D(i-1)= **POWER\_ON** & Ci=**HIGH\_LEVEL**，Ai = **SENSOR\_CONDUCT**
* if D(i-1)= **POWER\_OFF** & Ci=**LOW\_LEVEL**，keeps Ai as last status: Ai = Ai(t-1)
* if D(i-1)= **POWER\_OFF** & Ci=**HIGH\_LEVEL**，Ai = **SENSOR\_WRONG**

根据上述三个传感器的判断结果，判断编码里程计的状态，条件如下：

* A1/2/3中有任意一个为**SENSOR\_WRONG**，则编码里程计错误，设置本中断的[OdometerImm](#OdometerImm).[TestResult](#TestResult)为**TEST\_INCONSISTENT**；
* A1/2/3全都为**SENSOR\_CONDUCT**状态，则编码里程计错误，设置本中断的[OdometerImm](#OdometerImm).[TestResult](#TestResult)为**TEST\_INCONSISTENT**；
* A1/2/3全都为**SENSOR\_BLOCKED**状态，则编码里程计错误，设置本中断的[OdometerImm](#OdometerImm).[TestResult](#TestResult)为**TEST\_INCONSISTENT**；
* A1/2/3全都与上次中断中的A1/2/3状态一致，则认为里程计所在车轴静止，设置本中断的[OdometerImm](#OdometerImm).[TestResult](#TestResult)为**TEST\_STOPPING**；
* 如果中断在**TEST\_STOPPING**状态超过**SENSOR\_TEST\_IMMOBILE\_THRESHOLD**时间，则认为里程计完全静止，ATP设置此中断的[OdometerImm](#OdometerImm).[TestResult](#TestResult)为**TEST\_IMMOBILE**。
* A1/2/3中有任意一个的状态与上次中断中的状态不一致，则认为里程计所在车轴移动，设置本中断的[OdometerImm](#OdometerImm).[TestResult](#TestResult)为**TEST\_FLOATING**。

Based on the above three sensors’ status A1/2/3, ATP determines the status of the odometer as following conditions:

* If any one of A1/2/3 is **SENSOR\_WRONG**, then ATP consider the odometer as error in this interrupt and set the [OdometerImm](#OdometerImm).[TestResult](#TestResult)as **TEST\_INCONSISTENT**;
* If all of A1/2/3 are **SENSOR\_CONDUCT**, then ATP consider the odometer as error in this interrupt and set the [OdometerImm](#OdometerImm).[TestResult](#TestResult)as **TEST\_INCONSISTENT;**
* If all of A1/2/3 are **SENSOR\_BLOCKED**, then ATP consider the odometer as error in this interrupt and set the [OdometerImm](#OdometerImm).[TestResult](#TestResult)as **TEST\_INCONSISTENT**;
* If all of A1/2/3 are as same as the result at last interrupt respectively, then ATP consider the odometer as stop in this interrupt and set the [OdometerImm](#OdometerImm).[TestResult](#TestResult)as **TEST\_STOPPING**;
* If the **TEST\_STOPPING** has lasted more than **SENSOR\_TEST\_IMMOBILE\_THRESHOLD**, the ATP consider the odometer standstill, and set [OdometerImm](#OdometerImm).[TestResult](#TestResult)as **TEST\_IMMOBILE**;
* If any one of A1/2/3 is different with the result at last interrupt, then ATP consider the odometer rolling, and set the [OdometerImm](#OdometerImm).[TestResult](#TestResult)as **TEST\_FLOATING**.

当检测到**TEST\_INCONSISTENT**或者**TEST\_FLOATING**时，停止传感器测试，设置当前中断的[OdometerImm](#OdometerImm).[SensorTesting](#SensorTesting)标志为**False**，并设置本周期的[SensorTestFlag](#SensorTestFlag)为**False**。

When the odometer test result is either **TEST\_INCONSISTENT** or **TEST\_FLOATING**, ATP shall stop the sensor test and set [OdometerImm](#OdometerImm).[SensorTesting](#SensorTesting) as **False** for this interrupt, and set [SensorTestFlag](#SensorTestFlag) as **False** for this cycle.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149], [iTC\_CC-SyAD-0958], [iTC\_CC\_ATP\_SwHA-0035], [iTC\_CC\_ATP\_SwHA-0058]

[End]

**NOTES：**

由于设计限制，只有VPB-2板的上模块向传感器发送D1/2/3测试序列，同时将该测试序列转发给VPB-2板的下模块，供VLE-2板的CPU2读取。VPB-2板上下模块对于传感器测试结果C1/2/3的处理是一致的。

Because of the design restriction, only the FPGA1 in the VPB-2 board can send the testing sequence D1/2/3 to the sensor, and meanwhile the FPGA1 will forward the info to the FPGA2 in the VPB-2 so that the CPU2 in the VLE-2 can read. Both of the FPGA in the VPB-2 board will have the same process for the sensor testing result C1/2/3.

[iTC\_CC\_ATP-SwRS-0116]

在指定时刻**T\_LOCK\_ODOMETER**，锁存一个主周期所有中断中的[OdometerImm](#OdometerImm)到数组LockedOdometer[**ATP\_INTERRUPT\_NB** ]中，其下标为所在中断的[ImmediateNb](#ImmediateNb)；并使用LockedBeaconMsgReady，LockedTopLocCounter和LockedBeaconMsgByte分别锁存BeaconMsgReady，TopLocCounter和BeaconMsgByte的值供主任务使用。

In the specific **T\_LOCK\_ODOMETER**, the [OdometerImm](#OdometerImm) of all intervals need to be saved into [LockedOdometer](#LockedOdometer)[ATP\_INTERRUPT\_NB] with the index as [ImmediateNb](#ImmediateNb); The value of [BeaconMsgReady](#BeaconMsgReady), [TopLocCounter](#TopLocCounter) and [BeaconMsgByte](#BeaconMsgByte) should be recorded by using [LockedBeaconMsgReady](#LockedBeaconMsgReady), [LockedTopLocCounter](#LockedTopLocCounter) and [LockedBeaconMsgByte](#LockedBeaconMsgByte)for the main task.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0130]

[End]

[iTC\_CC\_ATP-SwRS-0590]

IdenticalLockedOdometer，上下CPU模块的ATP应当交互上周期中断中锁存的VPB信息，并遵循以下规则进行同步：

* 对于[CogCounter](#CogCounter)信息：
* 相同中断中若CogCounter不同，则取较大的作为同步后该中断的结果；
* 里程计齿号与齿数取值相同CPU的值。
* 对于[TopLocValid](#TopLocValid)信息：
* ATP应检查top-loc发生时VPB锁存的门闩寄存器锁存值[CalCogCounter](#CalCogCounter)是否在该中断的[CogCounter](#CogCounter)和上个中断的[CogCounter](#CogCounter)之间，若不在上述范围之间，则认为top-loc无效；
* 如果上下模块相差1个中断，则取前一个中断作为计算[CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc)的依据，而后一个中断作为计算[CogPositionAfterTopLoc](#CogPositionAfterTopLoc)的依据；
* 如果两个[TopLocValid](#TopLocValid)相差超过1个中断，则ATP认为该top-loc无效。

The ATP software in different CPUs shall synchronize the information get from the VPB board, with following rules:

* For [CogCounter](#CogCounter):
* If the [CogCounter](#CogCounter) read by two CPUs are different at the same interrupt, ATP shall take the large one as the result;
* ATP shall use the CPU's CogCode as same as [CogCounter](#CogCounter).
* For [TopLocValid](#TopLocValid):
* Only the cog counter latched by TOPLOC is between the before and after cog counter , TOPLOC is considered valid
* If the top-loc happened in adjacent interrupt between two CPUs, the former one shall use to calculate the [CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc), and the latter to calculate the [CogPositionAfterTopLoc](#CogPositionAfterTopLoc);
* If the top-loc difference are more than one interrupt, ATP shall consider it as invalid.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source =[iTC\_CC\_ATP\_SwHA-0232], [iTC\_CC-SyAD-1438]

[End]

#### Beacon message

当信标天线经过线路上信标上方时，会产生top-loc信号并送给VPB板。当ATP读到该信号时，记录此时刻的里程计齿数，并获取、解析及验证收到的信标消息。为防止VPB未能正确刷新信标消息寄存器里的值，ATP软件每周期将生成随机数并写给VPB，尤其将该数附加到信标消息中，作为收到该信标的时间信息。

When the Beacon antenna passes the upside of beacon, it will generate the top-loc signal and send it to VPB board. While the ATP gets this signal, it will lock the odometer cog counter at this moment, and will obtain, parse and check the beacon message.

[iTC\_CC\_ATP-SwRS-0120]

BeaconChecksumFailure，判断信标消息校核字是否正确。

* 主任务中，如果发现中断中[LockedBeaconMsgReady](#LockedBeaconMsgReady)为**True**，则需对[LockedBeaconMsgByte](#LockedBeaconMsgByte)信息进行校验，包括根据上周期或上上周期的ATCkey检测信标消息实时性，并计算信标的SACEM校核字。
* 如果校验错误，则设置 [BeaconChecksumFailure](#BeaconChecksumFailure)为**True**
* 如果校验正确，则设置 [BeaconChecksumFailure](#BeaconChecksumFailure)为**False**。
* 如果本周期[LockedBeaconMsgReady](#LockedBeaconMsgReady)为**False**，则设置[BeaconChecksumFailure](#BeaconChecksumFailure)为**False**。

[BeaconChecksumFailure](#BeaconChecksumFailure) judges whether the checksum of beacon message is correct or not.

* In the main task, if [LockedBeaconMsgReady](#LockedBeaconMsgReady) is **True**, [LockedBeaconMsgByte](#LockedBeaconMsgByte)need to be detected, including validity of [ATCkey](#ATCkey) and calculation of the SACEM checksum of beacon.
* ATP shall reject each beacon message which vital checksum is corrupted.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0164], [iTC\_CC-SyAD-0173], [iTC\_CC-SyAD-0957], [iTC\_CC-SyAD-0969], [iTC\_CC\_ATP\_SwHA-0036], [iTC\_CC\_VLE-2-VPB-2-SyID-0028]

[End]

[iTC\_CC\_ATP-SwRS-0117]

在ATP主任务中，如果中断中的[LockedBeaconMsgReady](#LockedBeaconMsgReady)状态为**True**，且[BeaconChecksumFailure](#BeaconChecksumFailure)为**False**，则设置BeaconMessageReceive为**True**；否则令其为**False**。其中，如果ATP在一个周期中收到多于一个信标时，仅处理最后一个信标，据此更新BeaconMessageReceive。

In the main task of ATP, if the status of [LockedBeaconMsgReady](#LockedBeaconMsgReady) in the interrupt is **True**, and the [BeaconChecksumFailure](#BeaconChecksumFailure) is **False**, ATP shall set the [BeaconMessageReceive](#BeaconMessageReceive) as **True**; and vice versa. If ATP receives more than one beacons in this cycle, it will deal with the last beacon and based on this data to update the [BeaconMessageReceive](#BeaconMessageReceive).

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0171], [iTC\_CC-SyAD-0164]

[End]

[iTC\_CC\_ATP-SwRS-0118]

BeaconCount，ATP记录从上电开始，到当前周期共收到多少次Top-loc信号。

[BeaconCount](#BeaconCount) represents the accumulated number of received Top-loc signal from power on to current cycle.

if (Initialization)

[BeaconCount](#BeaconCount) = 0

else:

[BeaconCount](#BeaconCount) = [LockedTopLocCounter](#LockedTopLocCounter)(k) + [BeaconCount](#BeaconCount)(k-1)

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0408]

[End]

[iTC\_CC\_ATP-SwRS-0119]

如果本周期[BeaconMessageReceive](#BeaconMessageReceive)为**True**，则解析锁存的[LockedBeaconMsgByte](#LockedBeaconMsgByte)数组，生成BeaconMessage信息，其结构为ST\_BEACON\_MSG：

If the [BeaconMessageReceive](#BeaconMessageReceive) is **True**, ATP shall parse the value of [LockedBeaconMsgByte](#LockedBeaconMsgByte) and generate BeaconMessage with structure as ST\_BEACON\_MSG:

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_BEACON\_MSG | |  |  |
|  | ID | NUMERIC\_32 | 信标标识 |
|  | Variants[**MAX\_BM\_VARIANT\_NB**] | BOOLEAN | 变量状态 |
|  | DefaultMessage | BOOLEAN | 是否默认消息 |
|  | BlockModeVariantAvailable | BOOLEAN | 所带变量是否有效 |

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0171]

[End]

[iTC\_CC\_ATP-SwRS-0121]

如果本周期[BeaconMessageReceive](#BeaconMessageReceive)为**True**，则从[LockedBeaconMsgByte](#LockedBeaconMsgByte)中获取信标ID，设置BeaconMessage.ID；其他情况保持不变。

If the status of [BeaconMessageReceive](#BeaconMessageReceive) is **True**, the [BeaconMessage](#BeaconMessage).ID is obtained by [LockedBeaconMsgByte](#LockedBeaconMsgByte); Otherwise, keep it unchanged.

if (Initialization)

[BeaconMessage](#BeaconMessage).ID = 0

elif ([BeaconMessageReceive](#BeaconMessageReceive)(k))

[BeaconMessage](#BeaconMessage).ID = [LockedBeaconMsgByte](#LockedBeaconMsgByte)[**BEACON\_ID\_BITS**]

else:

[BeaconMessage](#BeaconMessage).ID = [BeaconMessage](#BeaconMessage).ID(k-1)

其中**BEACON\_ID\_BITS**表示[REF4]中定义的信标消息中表示信标ID的位数。

**BEACON\_ID\_BITS** represents the index of beacon ID in the beacon message defined in [REF4].

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0175]

[End]

[iTC\_CC\_ATP-SwRS-0122]

如果本周期[BeaconMessageReceive](#BeaconMessageReceive)为**True**，则从[LockedBeaconMsgByte](#LockedBeaconMsgByte)中获取变量数据，设置数组BeaconMessage.Variants[**MAX\_BM\_VARIANT\_NB**]；若本周期未读到新的信标则保持不变。

If the [BeaconMessageReceive](#BeaconMessageReceive) is **True**, the variants is come from [LockedBeaconMsgByte](#LockedBeaconMsgByte)and ATP set as [BeaconMessage](#BeaconMessage).Variants[**MAX\_BM\_VARIANT\_NB**]; if there is no beacon read at the end of cycle, there is no changes.

if (Initialization)

[BeaconMessage](#BeaconMessage).Variants = {0,..,0}

elif ([BeaconMessageReceive](#BeaconMessageReceive)(k))

[BeaconMessage](#BeaconMessage).Variants(k)

= {[LockedBeaconMsgByte](#LockedBeaconMsgByte)[**BM\_VARIANTS\_BIT\_0**],

...,

[LockedBeaconMsgByte](#LockedBeaconMsgByte)[**BM\_VARIANTS\_BIT\_15**]}

else:

[BeaconMessage](#BeaconMessage).Variants = [BeaconMessage](#BeaconMessage).Variants(k-1)

其中**BM\_VARIANTS\_BIT\_0**...**BM\_VARIANTS\_BIT\_15**表示[REF4]中定义的信标消息中表示BM信标变量的位数。

**BM\_VARIANTS\_BIT\_0**...**BM\_VARIANTS\_BIT\_15** represents the index of BM beacon variants defined in [REF4].

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0163], [iTC\_CC-SyAD-0176]

[End]

[iTC\_CC\_ATP-SwRS-0123]

如果本周期[BeaconMessageReceive](#BeaconMessageReceive)为**True**，则从[LockedBeaconMsgByte](#LockedBeaconMsgByte)中判断是否默认消息，设置[BeaconMessage](#BeaconMessage).[DefaultMessage](#DefaultMessage)；若本周期未读到新的信标则保持不变。

If the [BeaconMessageReceive](#BeaconMessageReceive) is **True**, the default message is judged by [LockedBeaconMsgByte](#LockedBeaconMsgByte)and ATP set the [BeaconMessage](#BeaconMessage).[DefaultMessage](#DefaultMessage); if there is no new beacon read, it keeps unchanged.

if (Initialization)

[BeaconMessage](#BeaconMessage).[DefaultMessage](#DefaultMessage) = **False**

elif ([BeaconMessageReceive](#BeaconMessageReceive)(k))

[BeaconMessage](#BeaconMessage).[DefaultMessage](#DefaultMessage)(k)

= [LockedBeaconMsgByte](#LockedBeaconMsgByte)[**DEFAULT\_MESSAGE\_BI**T]

else:

[BeaconMessage](#BeaconMessage).[DefaultMessage](#DefaultMessage) = [BeaconMessage](#BeaconMessage).[DefaultMessage](#DefaultMessage)(k-1)

其中**DEFAULT\_MESSAGE\_BIT**表示[REF4]中定义的信标消息中表示信标是否为默认消息的位数。

**DEFAULT\_MESSAGE\_BIT** represents the index of beacon that judges default message, which defined in the [REF4].

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0170]

[End]

[iTC\_CC\_ATP-SwRS-0124]

如果本周期[BeaconMessageReceive](#BeaconMessageReceive)为**True**，则从[LockedBeaconMsgByte](#LockedBeaconMsgByte)中判断变量是否可用信息，设置[BeaconMessage](#BeaconMessage).[BlockModeVariantAvailable](#BlockModeVariantAvailable)；若本周期未读到新的信标则保持不变。

If the [BeaconMessageReceive](#BeaconMessageReceive) is **True**, it is feasible to judge whether the variants are available through [LockedBeaconMsgByte](#LockedBeaconMsgByte) and ATP set as [BeaconMessage](#BeaconMessage).[BlockModeVariantAvailable](#BlockModeVariantAvailable); If there is no new beacon read, it keeps invariable.

if (Initialization)

[BeaconMessage](#BeaconMessage).[BlockModeVariantAvailable](#BlockModeVariantAvailable) = **False**

elif ([BeaconMessageReceive](#BeaconMessageReceive)(k))

[BeaconMessage](#BeaconMessage).[BlockModeVariantAvailable](#BlockModeVariantAvailable)(k)

= [LockedBeaconMsgByte](#LockedBeaconMsgByte)[**BLOCK\_MODE\_VARIANT\_AVAILABLE\_BIT**]

else:

[BeaconMessage](#BeaconMessage).[BlockModeVariantAvailable](#BlockModeVariantAvailable)(k)

= [BeaconMessage](#BeaconMessage).[BlockModeVariantAvailable](#BlockModeVariantAvailable)(k-1)

其中**BLOCK\_MODE\_VARIANT\_AVAILABLE\_BIT**表示[REF4]中定义的信标消息中表示信标所带变量是否可用的位数。

**BLOCK\_MODE\_VARIANT\_AVAILABLE\_BIT** stands for the index of the beacon variants in the beacon message defined in [REF4].

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0170]

[End]

[iTC\_CC\_ATP-SwRS-0049]

车载ATP每周期计算得到伪随机数ATCkey，并将其写入VPB-2板相应寄存器，用于区分VPB-2消息的实时性。

At each cycle, ATP shall provide to beacon device the [ATCkey](#ATCkey) in order to be able to control message freshness.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0165], [iTC\_CC\_ATP\_SwHA-0175], [iTC\_CC-SyAD-0064], [iTC\_CC\_VLE-2-VPB-2-SyID-0028]

[End]

[iTC\_CC\_ATP-SwRS-0767]

ATP应读取VPB板的[CBKWrite](#CBKWrite).RadarReg信息，供维护诊断使用。

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source= [iTC\_CC\_VLE-2-VPB-2-SyID-0015], [iTC\_CC\_VLE-2-VPB-2-SyID-0029]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [ATCkey](#ATCkey) | √ | √ | NUMERIC\_32 |
| [BeaconCount](#BeaconCount) | × | √ | NUMERIC\_32 |
| [BeaconMessage](#BeaconMessage) | √ | √ | ST\_BEACON\_MSG |
| [BeaconMessageReceive](#BeaconMessageReceive) | √ | √ |  |
| [IdenticalLockedOdometer](#IdenticalLockedOdometer) | √ | √ |  |
| [LockedBeaconMsgByte](#LockedBeaconMsgByte) | √ | √ | NUMERIC\_32 |

## F22-Monitor the Odometer

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [CoreId](#CoreId) | Internal | F11-Acquire Configuration Data |
| [IdenticalLockedOdometer](#IdenticalLockedOdometer) | Internal | F21-Manage Interface with |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [CompCogCode](#CompCogCode) | √ | × | NUMERIC\_32 |
| [MaxCountCogsRunInCycleExceeded](#MaxCountCogsRunInCycleExceeded) | √ | √ | BOOLEAN |
| [SensorSequenceDetected\_1](#SensorSequenceDetected_1) | √ | √ | BOOLEAN |
| [SensorSequenceDetected\_2](#SensorSequenceDetected_2) | √ | √ | BOOLEAN |
| [SensorSequenceDetected\_3](#SensorSequenceDetected_3) | √ | √ | BOOLEAN |

### Processing

ATP软件每周期在主任务中，根据上周期各中断锁存的编码里程计信息，对里程计所在车轮的动力学参数进行计算。即根据寄存器读值，判断本周期转过的齿数；如果未检测到里程计转动，则需进行传感器测试的判断，并据此检测车轮是否完全静止。根据编码里程计的设计，在传感器测试中不可能发生三路全导通或者堵塞的状态。

In every cycle of main task, according to coded odometer information of each interrupt latch in the last cycle, ATP software shall calculate the kinematic parameters of the wheel. That is to say, ATP shall judge the cog numbers in this cycle according to the value in the register. If ATP did not detect the move of odometer, it will check whether the wheel is completely static based on the sensor testing. According to the design of coded odometer, it is impossible that three sensors are in conduct or blocked at the same time.

[iTC\_CC\_ATP-SwRS-0592]

CompCogCode，ATP软件需要根据编码里程计的码盘特性和旋转方向，计算8个比特的期望齿号值。

* 当里程计初始化成功时，设置CompCogCode为初始的CogCode。
* 此后，对中断中转过的每个齿：
* 如果相邻中断齿数递增，期望齿号由高位向低位右移1个比特，将新的比特C4array[C4ArrayIndex]放在最高位，并更新C4ArrayIndex。
* 如果相邻中断齿数递减，期望齿号由低位向高位左移1个比特，将新的比特C4array[C4ArrayIndex]放在最低位，并更新C4ArrayIndex。

其中，C4ArrayIndex为当前对应的齿数索引，取值为0~99。C4array[C4ArrayIndex]为当前齿数对应码盘的通堵状态，1表示导通，0表示堵住，详见[REF4]。

The ATP software needs to calculate the expected cog code with 8 bits, according to the encoding characteristic of the disc code and the direction of odometer rotation.

* When the odometer initialization, the expected [CompCogCode](#CompCogCode) shall be set as initial CogCode;
* Since then, for one cog rotated in interrupt, the corresponded bit shall be shift as following rules:
* if the cog increased in adjacent interrupts, the [CompCogCode](#CompCogCode) shall be shift a bit toward right from high to low; shift out the lowest one and set the new highest bit as C4array[C4ArrayIndex], and update C4ArrayIndex.
* otherwise, if the cog decreased, the [CompCogCode](#CompCogCode) shall be shift a bit toward left from low to high; shift out the highest bit and set the C4array[C4ArrayIndex] as the new lowest one, and update C4ArrayIndex accordingly.

In which, C4ArrayIndex is the current cog index, ranging from 0 to 99. C4array[C4ArrayIndex] is the array of disc codes, "1" meaning conduction and "0" indicating blocked, for details see [REF4].

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0013]

[End]

[iTC\_CC\_ATP-SwRS-0164]

OdometerCogPositionReady，根据[IdenticalLockedOdometer](#IdenticalLockedOdometer)中锁存各中断的CogCode和ATP主任务计算的期望齿号[CompCogCode](#CompCogCode)是否匹配，判断里程计齿数齿号的可用OdometerCogPositionReady状态。

* 初始化时，设置OdometerCogPositionReady为**False**。
* 否则，如果之前OdometerCogPositionReady为**False**，则只有当ATP检测到车轮反转或者停转（[WheelFilteredStopped](#WheelFilteredStopped)）后重新转动，使得里程计朝同一个方向连续转过8个齿后，初始化齿数和齿号的匹配关系，并设置OdometerCogPositionReady为**True**。
* 否则，如果本周期某中断中的期望齿号CompCogCode与读到的齿号CogCode不相等，则设置OdometerCogPositionReady为**False**
* 其他情况，保持OdometerCogPositionReady不变。

ATP determines the odometer position ready according to the matching of the [CompCogCode](#CompCogCode) and CogCode locked in each interrupt.

* In initialization, the OdometerCogPositonReady shall be **False**;
* Or else:, if the OdometerCogPositonReady was **False**, then only after the odometer rotated reversely or [WheelFilteredStopped](#WheelFilteredStopped) and re-turned continuous toward the same direction after 8 cogs, ATP shall re-initialize the Counter-Code matching relation and set OdometerCogPositonReady as **True**;
* Or else:, if CogCode is different with [CompCogCode](#CompCogCode) in one of interrupt of the cycle, ATP shall set OdometerCogPositonReady as **False**;
* Otherwise, ATP keep OdometerCogPositonReady unchanging.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0013], [iTC\_CC-SyAD-0960], [iTC\_CC\_ATP\_SwHA-0053]

[End]

[iTC\_CC\_ATP-SwRS-0165]

TeethCounter，ATP根据[IdenticalLockedOdometer](#IdenticalLockedOdometer)中锁存的最后一个中断的[CogCounter](#CogCounter)变化值，更新TeethCounter，作为主任务使用的里程计齿数值。TeethCounter的计算应考虑里程计安装方向和CogCounter的寄存器取值范围。

[TeethCounter](#TeethCounter) used as the odometer cog value in one deferred task, which is the difference of the [CogCounter](#CogCounter) in the last interrupt of adjacent cycle. The calculation of the [TeethCounter](#TeethCounter) shall consider the installation direction of the odometer and the register range of the [CogCounter](#CogCounter).

[TeethCounter](#TeethCounter)(k)

= [TeethCounter](#TeethCounter)(k-1)

+ ([IdenticalLockedOdometer](#IdenticalLockedOdometer)[ATP\_INTERRUPT\_NB - 1].[CogCounter](#CogCounter)(k)

- [IdenticalLockedOdometer](#IdenticalLockedOdometer)[ATP\_INTERRUPT\_NB - 1].[CogCounter](#CogCounter)(k-1))

\* [ATPsetting](#ATPsetting).CCcoreOdoCogIncreasing[[CoreId](#CoreId)]

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0130], [iTC\_CC-SyAD-0135], [iTC\_CC\_ATP\_SwHA-0181]

[End]

**NOTES**:

[TeethCounter](#TeethCounter)是有符号值。如果[TeethCounter](#TeethCounter)大于0，则表示里程计相对于初始位置向列车**END\_1**方向转动；反之如果小于0，则表示里程计相对于初始位置向列车**END\_2**方向转动。

[TeethCounter](#TeethCounter) is a signed value. If [TeethCounter](#TeethCounter) greater than 0, then means the odometer rotating toward to the train **END\_1** direction; other hand, if it less than 0, then means the odometer rotating toward to the **END\_2**.

[iTC\_CC\_ATP-SwRS-0166]

CogPositionBeforeTopLoc，CogPositionAfterTopLoc，如果本周期读到信标，则通过[IdenticalLockedOdometer](#IdenticalLockedOdometer)计算读到信标瞬间的里程计齿数信息：

* 使用Top-loc发生的前一个中断的[CogCounter](#CogCounter)来更新CogPositionBeforeTopLoc；
* 使用Top-loc发生时中断的[CogCounter](#CogCounter)来更新CogPositionAfterTopLoc；

其他情况，CogPositionBeforeTopLoc和CogPositionAfterTopLoc保持不变。

If a beacon with top-loc received in this cycle, ATP shall record the cog position of the interrupt when and just before the top-loc happen:

* [CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc), the [CogCounter](#CogCounter) in the interrupt just before the top-loc happen;
* [CogPositionAfterTopLoc](#CogPositionAfterTopLoc), the [CogCounter](#CogCounter) in the interrupt when the top-loc happen.

[CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc)(k)

= [TeethCounter](#TeethCounter)(k-1)

+ (([IdenticalLockedOdometer](#IdenticalLockedOdometer)[i-1].[CogCounter](#CogCounter)(k)

- [IdenticalLockedOdometer](#IdenticalLockedOdometer)[**ATP\_INTERRUPT\_NB**-1].[CogCounter](#CogCounter)(k-1))

\* [ATPsetting](#ATPsetting).CCcoreOdoCogIncreasing[[CoreId](#CoreId)]))

[CogPositionAfterTopLoc](#CogPositionAfterTopLoc)(k)

= [TeethCounter](#TeethCounter)(k-1)

+ (([IdenticalLockedOdometer](#IdenticalLockedOdometer)[i].[CogCounter](#CogCounter)(k)

- [IdenticalLockedOdometer](#IdenticalLockedOdometer)[**ATP\_INTERRUPT\_NB**-1].[CogCounter](#CogCounter)(k-1))

\* [ATPsetting](#ATPsetting).CCcoreOdoCogIncreasing[[CoreId](#CoreId)]))

其中i表示锁存收到Top-loc信号的那个中断。如果上下CPU收到Top-loc相差1个中断，则使用较早的的中断作为计算CogPositionBeforeTopLoc的依据，而较迟的那个中断作为计算CogPositionAfterTopLoc的依据。

Which, i means the interrupt received top-loc signal.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0197], [iTC\_CC\_ATP\_SwHA-0059]

[End]

[iTC\_CC\_ATP-SwRS-0167]

SensorTestPerformed，当主任务通过锁存的[IdenticalLockedOdometer](#IdenticalLockedOdometer)数组发现本周期所有的中断均正在对传感器进行测试时，输出SensorTestPerformed为**True**。

否则，输出SensorTestPerformed为**False**。

If all interrupts in one cycle are sensors testing, ATP shall set [SensorTestPerformed](#SensorTestPerformed). Otherwise, set [SensorTestPerformed](#SensorTestPerformed) as **False**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149]

[End]

[iTC\_CC\_ATP-SwRS-0464]

SensorSequenceDetected\_1，SensorSequenceDetected\_2，SensorSequenceDetected\_3，

如果本周期[SensorTestPerformed](#SensorTestPerformed)为**True**，即中断在进行传感器测试，需分别判断三路传感器的导通状态[SensorSequenceDetected\_1](#SensorSequenceDetected_1)，[SensorSequenceDetected\_2](#SensorSequenceDetected_2)和SensorSequenceDetected \_3：

* 如果该路传感器在本周期所有中断的测试结果均为**SENSOR\_CONDUCT**，则设置相应状态为**True**；
* 否则，设置相应传感器状态为**False**。

如果本周期[SensorTestPerformed](#SensorTestPerformed)为**False**，则设置三路传感器状态均为**False**。

If sensors testing performed in this cycle, ATP shall determine the conduction state of each sensor:

* If all test results of every interrupts for this sensor are **SENSOR\_CONDUCT**, ATP shall set sensor sequence detected for this sensor;
* Otherwise, does not set this sensor sequence detected.

If sensors testing do not perform, ATP does not set any sensor sequence detected.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149]

[End]

[iTC\_CC\_ATP-SwRS-0168]

UnconsistentSensorTest，通过检查C1/2/3传感器的一致性，判断里程计故障。

* 如果在传感器测试中，任意一次中断中的传感器测试的结果为C1/2/3三路全为**SENSOR\_CONDUCT**，或三路全为**SENSOR\_BLOCKED，**或者任意一路为**SENSOR\_WRONG**，则设置本周期[UnconsistentSensorTest](#UnconsistentSensorTest)为**True**；
* 或者，在非传感器测试时，如果某中断的C1/2/3三路状态相同（同为导通或同为堵塞），也应设置本周期[UnconsistentSensorTest](#UnconsistentSensorTest)为**True**；
* 否则，设置[UnconsistentSensorTest](#UnconsistentSensorTest)为**False**。

ATP shall check consistency of sensors whether sensor testing performed or not. Sensors testing result shall declare inconsistent at cycle k ([UnconsistentSensorTest](#UnconsistentSensorTest)) if the following conditions are fulfilled:

* Sensors test done at cycle k, and no sequence has been detected on any of the three sensors C1, C2, C3 (**SENSOR\_BLOCKED**)
* Or at cycle k, the expected sequence is detected on all three sensors C1, C2, C3 (**SENSOR\_CONDUCT**).
* Or any of the three sensors is tested as error (**SENSOR\_WRONG**).

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149], [iTC\_CC\_ATP\_SwHA-0035], [iTC\_CC\_ATP\_SwHA-0057]

[End]

[iTC\_CC\_ATP-SwRS-0171]

WheelStopped，如果当前在进行传感器测试，且任一中断中未发生三路全通或全堵错误，且一个周期所有中断内三路传感器的导通状态都与上周期的结果相同时，输出WheelStopped为**True**。否则为**False**。

Wheel shall consider safely stopped [WheelStopped](#WheelStopped) at cycle k if the following conditions are fulfilled:

* sensors test has been performed,
* and at least one sensor out of three sensors C1, C2, C3 has detected expected sequence,
* and at least one sensor out of three sensors C1, C2, C3 has not detected expected sequence,
* and sensors test result combination on three sensors C1, C2, C3 has not changed between cycle k-1 and k.

if ([SensorTestPerformed](#SensorTestPerformed)(k) == **True**)

[WheelStopped](#WheelStopped)(k)

= (([UnconsistentSensorTest](#UnconsistentSensorTest)(k) == **False**)

and ([SensorSequenceDetected\_1](#SensorSequenceDetected_1) = [SensorSequenceDetected\_1](#SensorSequenceDetected_1)(k-1))

and ([SensorSequenceDetected\_2](#SensorSequenceDetected_2) = [SensorSequenceDetected\_2](#SensorSequenceDetected_2)(k-1))

and ([SensorSequenceDetected\_3](#SensorSequenceDetected_3) = [SensorSequenceDetected\_3](#SensorSequenceDetected_3)(k-1)))

else:

[WheelStopped](#WheelStopped) = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0150]

[End]

[iTC\_CC\_ATP-SwRS-0172]

WheelFilteredStopped，判断本周期车轮是否处于滤过停止状态，规则如下：

* 如果WheelFilteredStopped上周期为**False**，而本周期[WheelStopped](#WheelStopped)由**False**变为**True**，则认为本周期为**True**。
* 在此条件下，记录停车时的齿数LastStopCogPosition为当前齿数
* WheelFilteredStopped由**True**变为**False**的条件：
* 齿数移动超过1个齿

At cycle k, [WheelFilteredStopped](#WheelFilteredStopped) shall change from **False** to **True** on raising edge of [WheelStopped](#WheelStopped) information, That is, if:

* [WheelStopped](#WheelStopped) information was **False** at cycle k-1,
* and [WheelStopped](#WheelStopped) information was **True** at cycle k.
* and then:
* LastStopCogPosition is assigned to [TeethCounter](#TeethCounter),

At cycle k, [WheelFilteredStopped](#WheelFilteredStopped) shall change from **True** to **False**, according following expression:

* the cog moved more than one cog;

def [WheelFilteredStopped](#WheelFilteredStopped)(k):

if (not [WheelFilteredStopped](#WheelFilteredStopped)(k-1)

and not [WheelStopped](#WheelStopped)(k-1)

and [WheelStopped](#WheelStopped)(k)):

LastStopCogPosition = [TeethCounter](#TeethCounter)(k)

return **True**

elif ([WheelFilteredStopped](#WheelFilteredStopped)(k-1)

and not [UnconsistentSensorTest](#UnconsistentSensorTest)(k)

and [abs](#abs)([TeethCounter](#TeethCounter)(k) - LastStopCogPosition) <= 1):

return **True**

else:

return **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0151], [iTC\_CC-SyAD-0214]

[End]

[iTC\_CC\_ATP-SwRS-0173]

MaxCountCogsRunInCycleExceeded，里程计转过齿数不能超过周期最大值，也不能超过的相邻中断的最大值。

ATP shall detect whether the cog number counted in adjacent interrupt is greater than the default maximum cog number on cycle or on interrupt.

def [MaxCountCogsRunInCycleExceeded](#MaxCountCogsRunInCycleExceeded)(k):

if (abs([IdenticalLockedOdometer](#IdenticalLockedOdometer)[**ATP\_INTERRUPT\_NB**-1].CogCounter(k)

- [IdenticalLockedOdometer](#IdenticalLockedOdometer)[**ATP\_INTERRUPT\_NB**-1].CogCounter(k-1))

> ATPsetting.OdoMaxCogOnCycle):

return **True**

else:

for i in [range](#range)(**ATP\_INTERRUPT\_NB-1**):

if ([abs](#abs)([IdenticalLockedOdometer](#IdenticalLockedOdometer)[i].[CogCounter](#CogCounter)

- [IdenticalLockedOdometer](#IdenticalLockedOdometer)[i+1].[CogCounter](#CogCounter))> [ATPsetting](#ATPsetting).OdoMaxCogOnIntrrupt):

return **True**

else:

continue

return **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0149], [iTC\_CC-SyAD-1158], [iTC\_CC\_ATP\_SwHA-0054]

[End]

**NOTES**:

[ATPsetting](#ATPsetting).OdoMaxCogOnCycle，由CC离线工具将里程计允许最大速度，根据项目最大齿距转换的每周期允许转过的最大齿数；

[ATPsetting](#ATPsetting).OdoMaxCogOnInterrupt，由CC离线工具将里程计允许最大速度，根据项目最小齿距转换的每中断允许转过的最大齿数。

[iTC\_CC\_ATP-SwRS-0174]

WheelKinematicsInvalidForCogCount，如果ATP检测到某个中断的齿数转过最大值时，设置齿数计算错误。

If the calculated movement exceeds the default one, ATP shall set the wheel kinematics invalid.

[WheelKinematicsInvalidForCogCount](#WheelKinematicsInvalidForCogCount) = [MaxCountCogsRunInCycleExceeded](#MaxCountCogsRunInCycleExceeded)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149], [iTC\_CC-SyAD-0960], [iTC\_CC-SyAD-1158], [iTC\_CC\_ATP\_SwHA-0054]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [CogPositionAfterTopLoc](#CogPositionAfterTopLoc) | √ | √ | NUMERIC\_32 |
| [CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc) | √ | √ | NUMERIC\_32 |
| [OdometerCogPositionReady](#OdometerCogPositionReady) | √ | √ | BOOLEAN |
| [SensorTestPerformed](#SensorTestPerformed) | √ | √ | BOOLEAN |
| [TeethCounter](#TeethCounter) | √ | √ | NUMERIC\_32 |
| [UnconsistentSensorTest](#UnconsistentSensorTest) | √ | √ | BOOLEAN |
| [WheelFilteredStopped](#WheelFilteredStopped) | √ | √ | BOOLEAN |
| [WheelKinematicsInvalidForCogCount](#WheelKinematicsInvalidForCogCount) | √ | √ | BOOLEAN |
| [WheelStopped](#WheelStopped) | √ | √ | BOOLEAN |

## F23-Manage the Odometer State

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [CoreId](#CoreId) | Internal | F11-Acquire Configuration Data |
| [OdometerCogPositionReady](#OdometerCogPositionReady) | Internal | F22-Monitor the Odometer |
| [MaxCogCalibration](#MaxCogCalibration) | Internal | F28-Calibrate Wheel Movement |
| [MinCogCalibration](#MinCogCalibration) | Internal | F28-Calibrate Wheel Movement |
| [SensorTestPerformed](#SensorTestPerformed) | Internal | F22-Monitor the Odometer |
| [TeethCounter](#TeethCounter) | Internal | F22-Monitor the Odometer |
| [WheelFilteredStopped](#WheelFilteredStopped) | Internal | F22-Monitor the Odometer |
| [WheelKinematicsInvalidForCogCount](#WheelKinematicsInvalidForCogCount) | Internal | F22-Monitor the Odometer |
| [UnconsistentSensorTest](#UnconsistentSensorTest) | Internal | F22-Monitor the Odometer |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [InitializationTimer](#InitializationTimer) | √ | √ | NUMERIC\_32 |
| [NoCommunicationWithOdometer](#NoCommunicationWithOdometer) | √ | √ | BOOLEAN |
| [SensorTestContradiction](#SensorTestContradiction) | √ | √ | BOOLEAN |

### Processing

ATP软件每周期在主任务中，需根据对传感器测试结果以及里程计齿数齿号的匹配情况，判断编码里程计的状态OdometerState，并以此计算车轮最大最小位移WheelMaximumMovement和WheelMinimumMovement。

里程计状态分为如下几种，如Figure 5‑7所示：

* **NOT\_INITIALIZED**，ATP刚上电后，里程计还未初始化的状态。此时若非已进行传感器测试并检测到车轮完全静止，否则应当过估车轮位移确保安全。
* **WAITING\_COG\_POSITION\_CODE\_READY**，ATP刚上电后，从里程计开始转动到连续转过8个齿进行初始化的过程。此过程中应当过估车轮位移，确保安全。
* **INITIALIZED**，里程计经过初始化后正常工作的状态。在此状态下，如果检测到里程计齿数齿号相匹配或者经过传感器测试并检测到车轮完全静止，则使用里程计的读值计算车轮位移；否则，应当过估车轮位移确保安全。
* **INVALID**，里程计的无效状态。

In the main task of each cycle, ATP need to estimate the odometer state and calculate WheelMaximumMovement和WheelMinimumMovement, based on the sensor test results and the matching status between cog count and cog number of odometer. As shown in Figure 5‑7 there are several odometer state:

* **NOT\_INITIALIZED**, just after ATP powered up, the odometer is not initialized. At this moment, ATP should overvalue the wheel displacement to ensure the safety only when the sensor detected that the wheel is completely static.
* **WAITING\_COG\_POSITION\_CODE\_READY**, the odometer starts rolling and continues to roll eight cogs. During this process, ATP should over-estimate wheel displacement to ensure the safety.
* **INITIALIZED**, when odometer has initialized, it enters into the normal working status. If ATP detected the cog count and cog number is matching or the wheel is static, it can calculate the wheel displacement by using the odometer value, otherwise, it should over- estimate the displacement.
* **INVALID**, the odometer is in the invalid status.

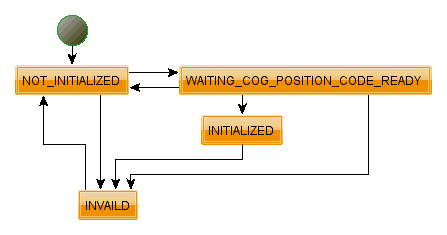


Figure 5‑7 Odometer state

#### State “NOT\_INITIALIZED”

[iTC\_CC\_ATP-SwRS-0175]

上电后里程计状态为**NOT\_INITIALIZED**。

From power-up, ATP shall consider that [OdometerState](#OdometerState) is "**NOT\_INITIALIZED** ".

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149]

[End]

[iTC\_CC\_ATP-SwRS-0176]

If [OdometerState](#OdometerState) is **NOT\_INITIALIZED** at cycle k, and if wheel detected stopped at cycle k, then [WheelMinimumMovement](#WheelMinimumMovement) and [WheelMaximumMovement](#WheelMaximumMovement) shall be set to zero.

if ([OdometerState](#OdometerState)(k) == **NOT\_INITIALIZED**)

if ([WheelFilteredStopped](#WheelFilteredStopped)(k) == **True**))

[WheelMinimumMovement](#WheelMinimumMovement) = 0

[WheelMaximumMovement](#WheelMaximumMovement) = 0

else:

[WheelMinimumMovement](#WheelMinimumMovement) = —[ATPsetting](#ATPsetting).MaxMotionPerCycle

[WheelMaximumMovement](#WheelMaximumMovement) = [ATPsetting](#ATPsetting).MaxMotionPerCycle

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0132], [iTC\_CC\_ATP\_SwHA-0060]

[End]

[iTC\_CC\_ATP-SwRS-0177]

里程计状态由**NOT\_INITIALIZED**变为**WAITING\_COG\_POSITION\_CODE\_READY**的条件是:

* 上周期在**NOT\_INITIALIZED**；
* 上周期在[WheelFilteredStopped](#WheelFilteredStopped)；
* 本周期未[WheelFilteredStopped](#WheelFilteredStopped)而且未检测到传感器测试失败

At cycle k, ATP shall consider that [OdometerState](#OdometerState) changes from **NOT\_INITIALIZED** to **WAITING\_COG\_POSITION\_CODE\_READY** if:

* a falling edge is detected on [[WheelFilteredStopped](#WheelFilteredStopped)](#WheelStopped) information,
* and sensors test is consistent at cycle k and was consistent at cycle k-1,

if (([OdometerState](#OdometerState)(k-1) = **NOT\_INITIALIZED**)

and (not [[WheelFilteredStopped](#WheelFilteredStopped)](#WheelStopped)(k) and not [UnconsistentSensorTest](#UnconsistentSensorTest)(k))

and ([[WheelFilteredStopped](#WheelFilteredStopped)](#WheelStopped)(k-1))

[OdometerState](#OdometerState) = **WAITING\_COG\_POSITION\_CODE\_READY**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149]

[End]

[iTC\_CC\_ATP-SwRS-0178]

若检测到传感器三路全通或全堵，则进入**INVALID**传感器无效

At cycle k, ATP shall consider that [OdometerState](#OdometerState) changes from **NOT\_INITIALIZED** to **INVALID**

* if sensors test is not consistent at cycle k.

if (([OdometerState](#OdometerState)(k-1) = **NOT\_INITIALIZED**)

**and** [UnconsistentSensorTest](#UnconsistentSensorTest)(k))

[OdometerState](#OdometerState) = **INVALID**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149], [iTC\_CC\_ATP\_SwHA-0057]

[End]

#### State “WAITING\_COG\_POSITION\_CODE\_READY”

[iTC\_CC\_ATP-SwRS-0179]

InitializationTimer，在**WAITING\_COG\_POSITION\_CODE\_READY**状态下累加初始化时间.

ATP shall accumulate the time for waiting cog position ready state.

if ([OdometerState](#OdometerState)(k-1) == **WAITING\_COG\_POSITION\_CODE\_READY**

and [OdometerState](#OdometerState)(k) == **WAITING\_COG\_POSITION\_CODE\_READY**)

[InitializationTimer](#InitializationTimer) = [InitializationTimer](#InitializationTimer)(k-1) + 1

elif ([OdometerState](#OdometerState)(k-1) != **WAITING\_COG\_POSITION\_CODE\_READY**

and [OdometerState](#OdometerState)(k) == **WAITING\_COG\_POSITION\_CODE\_READY**)

[InitializationTimer](#InitializationTimer) = 1

else:

[InitializationTimer](#InitializationTimer) = 0

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1160]

[End]

[iTC\_CC\_ATP-SwRS-0180]

由**WAITING\_COG\_POSITION\_CODE\_READY**转回**NOT\_INITIALIZED**状态的条件：

At cycle k, ATP shall consider that [OdometerState](#OdometerState) changes from **WAITING\_COG\_POSITION\_CODE\_READY** to **NOT\_INITIALIZED** if:

* wheel is detected stopped ([[WheelFilteredStopped](#WheelFilteredStopped)](#WheelStopped)),
* and cog position remains unknown (not [OdometerCogPositionReady](#OdometerCogPositionReady)),
* and there is no sensors test inconsistency,
* and time elapsed since last time [OdometerState](#OdometerState) was **NOT\_INITIALIZED** ([InitializationTimer](#InitializationTimer)) is strictly less than [ATPsetting](#ATPsetting).OdoInitTimeout

if ([OdometerState](#OdometerState)(k-1) = **WAITING\_COG\_POSITION\_CODE\_READY**)

and [[WheelFilteredStopped](#WheelFilteredStopped)](#WheelStopped)(k)

and not [OdometerCogPositionReady](#OdometerCogPositionReady)(k)

and not [UnconsistentSensorTest](#UnconsistentSensorTest)(k)

and ([InitializationTimer](#InitializationTimer)(k) < [ATPsetting](#ATPsetting).OdoInitTimeout)

[OdometerState](#OdometerState) = **NOT\_INITIALIZED**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0149], [iTC\_CC-SyAD-1160]

[End]

[iTC\_CC\_ATP-SwRS-0181]

由**WAITING\_COG\_POSITION\_CODE\_READY**转入**INITIALIZED**状态的条件：

At cycle k, ATP shall consider that [OdometerState](#OdometerState) changes from **WAITING\_COG\_POSITION\_CODE\_READY** to **INITIALIZED**

If:

* Cog position is safely known which means that wheel angular position is well-known;
* and there is no sensors test inconsistency;
* and time elapsed since last time [OdometerState](#OdometerState) was **NOT\_INITIALIZED** ([InitializationTimer](#InitializationTimer)) is strictly less than [ATPsetting](#ATPsetting).OdoInitTimeout.

if ([OdometerState](#OdometerState)(k-1) == **WAITING\_COG\_POSITION\_CODE\_READY**

and [OdometerCogPositionReady](#OdometerCogPositionReady)(k)

and not [UnconsistentSensorTest](#UnconsistentSensorTest)(k)

and ([InitializationTimer](#InitializationTimer)(k)< [ATPsetting](#ATPsetting).OdoInitTimeout))

[OdometerState](#OdometerState) = **INITIALIZED**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0138], [iTC\_CC-SyAD-1160], [iTC\_CC\_ATP\_SwHA-0061]

[End]

[iTC\_CC\_ATP-SwRS-0182]

由**WAITING\_COG\_POSITION\_CODE\_READY**转入**INVALID**的条件：

At cycle k, ATP shall consider that [OdometerState](#OdometerState) changes from **WAITING\_COG\_POSITION\_CODE\_READY** to **INVALID** if:

* sensors test inconsistency is detected,
* or time elapsed since last time [OdometerState](#OdometerState) was **NOT\_INITIALIZED** ([InitializationTimer](#InitializationTimer)) is more than or equal to the [ATPsetting](#ATPsetting).OdoInitTimeout

if ([OdometerState](#OdometerState)(k-1) == **WAITING\_COG\_POSITION\_CODE\_READY**)

and (([InitializationTimer](#InitializationTimer)(k)>= [ATPsetting](#ATPsetting).OdoInitTimeout)

or [UnconsistentSensorTest](#UnconsistentSensorTest)(k))

[OdometerState](#OdometerState) = **INVALID**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0149], [iTC\_CC-SyAD-1160]

[End]

[iTC\_CC\_ATP-SwRS-0183]

在里程计初始化阶段，ATP需根据当前车头激活方向和上周期位移的结果，对本周期位移进行过估处理。

When odometer is initializing, wheel movement shall be over and under estimated considering maximum acceleration per cycle according with the train front:

if ([OdometerState](#OdometerState)(k) == **WAITING\_COG\_POSITION\_CODE\_READY**)

if (([TrainFrontEnd](#TrainFrontEnd)(k-1) == **END\_2**) or ([NoUndetectableDanger\_2](#NoUndetectableDanger_2)(k-1) == **True**))

[WheelMinimumMovement](#WheelMinimumMovement)(k) = [WheelMinimumMovement](#WheelMinimumMovement)(k-1) + [ATPsetting](#ATPsetting).MaxMotionPerCycle

[WheelMaximumMovement](#WheelMaximumMovement)(k) = [WheelMaximumMovement](#WheelMaximumMovement)(k-1) - [ATPsetting](#ATPsetting).MaxMotionPerCycle

else:

[WheelMinimumMovement](#WheelMinimumMovement)(k) = [WheelMinimumMovement](#WheelMinimumMovement)(k-1) — [ATPsetting](#ATPsetting).MaxMotionPerCycle

[WheelMaximumMovement](#WheelMaximumMovement)(k) = [WheelMaximumMovement](#WheelMaximumMovement)(k-1) + [ATPsetting](#ATPsetting).MaxMotionPerCycle

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0132], [iTC\_CC\_ATP\_SwHA-0062]

[End]

#### State “INITIALIZED”

[iTC\_CC\_ATP-SwRS-0186]

当上周期里程计已在**INITIALIZED**状态，并满足以下条件之一时，里程计状态由**INITIALIZED**变为**INVALID**：

* 传感器测试检测出三路全通全堵；
* 或者，非停车状态，而且齿数齿号也不一致。

At cycle k, ATP shall consider that [OdometerState](#OdometerState) changes from **INITIALIZED** to **INVALID** if:

* [OdometerState](#OdometerState) was evaluated Initialized at cycle k-1,
* And:
* Sensors test result is inconsistent;
* Or neither wheel filtered stopped nor cog position ready.

if ([OdometerState](#OdometerState)(k-1) is **INITIALIZED**

and (([UnconsistentSensorTest](#UnconsistentSensorTest)(k) == **True**)

or (not [WheelFilteredStopped](#WheelFilteredStopped)(k)

and not [OdometerCogPositionReady](#OdometerCogPositionReady)(k)))):

[OdometerState](#OdometerState) = **INVALID**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149], [iTC\_CC-SyAD-0960], [iTC\_CC\_ATP\_SwHA-0063]

[End]

**NOTES**：

Wheel angular movement is the rotation movement observed on the wheel in cog count from cycle k-1 to k. Angular movement of the wheel can convert into a linear movement by taking into consideration of the uncertainty on wheel diameter measurement provided by calibration process. [WheelMinimumMovement](#WheelMinimumMovement) and [WheelMaximumMovement](#WheelMaximumMovement) represents respectively minimum and maximum curvilinear distance ran between cycle k-1and k by a reference point of the wheel located on the rolling circumference.

[iTC\_CC\_ATP-SwRS-0187]

在**INITIALIZED**状态，如果齿数齿号匹配，则计算车轮最大最小位移依据伪代码中的公式：

If motion and speed are available at cycle k, then wheel curvilinear movement calculates as follows:

if ([OdometerState](#OdometerState)(k) is **INITIALIZED**):

[WheelMinimumMovement](#WheelMinimumMovement)(k) = [MinCogCalibration](#MinCogCalibration)(k-1) \* ([TeethCounter](#TeethCounter)(k) — [TeethCounter](#TeethCounter)(k-1))

[WheelMaximumMovement](#WheelMaximumMovement)(k) = [MaxCogCalibration](#MaxCogCalibration)(k-1) \* ([TeethCounter](#TeethCounter)(k) — [TeethCounter](#TeethCounter)(k-1))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0130], [iTC\_CC-SyAD-0132], [iTC\_CC-SyAD-0135], [iTC\_CC\_ATP\_SwHA-0064]

[End]

**NOTES**:

对于车载ATP软件的位移，在齿数齿号匹配的状态下，无论ATP位于**END\_1**还是**END\_2**，也无论激活哪段车头，始终以**END\_1**方向为位移的正方向。即当位移大于0时，表示列车向**END\_1**端方向运行，反之则向**END\_2**端方向运行。

When odometer cog-counter-code matched, regardless of ATP in **END\_1** or **END\_2**, and no matter the activation of train front, the direction towards **END\_1** is always be set as the positive direction. That is, when the movement is greater than 0, indicating the direction of the train is running to **END\_1**, and vice versa to **END\_2**.

#### State “INVALID”

[iTC\_CC\_ATP-SwRS-0189]

在无效状态停车，并未检测到传感器错误，则能回到非初始化状态。

At cycle k, ATP shall consider that [OdometerState](#OdometerState) changes from **INVALID** to **NOT\_INITIALIZED** if:

* [OdometerState](#OdometerState) was evaluated Invalid at cycle k-1,
* and wheel is detected stopped ([[WheelFilteredStopped](#WheelFilteredStopped)](#WheelStopped)),
* and there is no sensors test inconsistency.

if ([OdometerState](#OdometerState)(k-1) == **INVALID**

and [[WheelFilteredStopped](#WheelFilteredStopped)](#WheelStopped)(k)

and not [UnconsistentSensorTest](#UnconsistentSensorTest)(k))

[OdometerState](#OdometerState) = **NOT\_INITIALIZED**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149]

[End]

[iTC\_CC\_ATP-SwRS-0578]

在里程计无效状态下，ATP直接使用测得值计算车轮位移（因为此时列车运动学失效，后续功能并不使用测得的列车车轮位移）。

In invalid status, ATP shall calculate wheel movement by using measured value of the odometer.

if ([OdometerState](#OdometerState)(k) == **INVALID**)

[WheelMinimumMovement](#WheelMinimumMovement)(k) = [MinCogCalibration](#MinCogCalibration)(k-1) \* ([TeethCounter](#TeethCounter)(k) — [TeethCounter](#TeethCounter)(k-1))

[WheelMaximumMovement](#WheelMaximumMovement)(k) = [MaxCogCalibration](#MaxCogCalibration)(k-1) \* ([TeethCounter](#TeethCounter)(k) — [TeethCounter](#TeethCounter)(k-1))

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0149]

[End]

#### Wheel kinematic calculation

[iTC\_CC\_ATP-SwRS-0170]

SensorTestContradiction，当里程计读数为0，但中断中却未进行传感器测试时，设置该变量为**True**，否则为**False**。

NoCommunicationWithOdometer，当SensorTestContradiction保持为**True**超过限定时间后，设置该值为真，表明中断中的传感器测试判断失败。

ATP shall invalidate wheel kinematic if minimum odometer motion is null and sensors test is not performed for more than [ATPsetting](#ATPsetting).OdoTestContradictionDuration.

def [SensorTestContradiction](#SensorTestContradiction)(k):

return (not [WheelFilteredStopped](#WheelFilteredStopped)(k)

and [TeethCounter](#TeethCounter)(k-1) == [TeethCounter](#TeethCounter)(k-2)

and not [SensorTestPerformed](#SensorTestPerformed)(k))

def [NoCommunicationWithOdometer](#NoCommunicationWithOdometer)(k):

if (Initialization

or not [SensorTestContradiction](#SensorTestContradiction)(k)):

SensorTestContradictionDuration = 0

return **False**

else:

SensorTestContradictionDuration = SensorTestContradictionDuration(k-1) + 1

if (SensorTestContradictionDuration > [ATPsetting](#ATPsetting).OdoTestContradictionDuration):

return **True**

else:

return False

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1161], [iTC\_CC\_ATP\_SwHA-0066]

[End]

[iTC\_CC\_ATP-SwRS-0190]

ValidWheelKinematic，车轮运动学特性有效.

Wheel kinematic is valid if odometer is valid, the calculated motion is not greater than the default value, and there is communication with odometer.

[ValidWheelKinematic](#ValidWheelKinematic)(k)

= (([OdometerState](#OdometerState)(k) != **INVALID**)

and (not [WheelKinematicsInvalidForCogCount](#WheelKinematicsInvalidForCogCount)(k))

and (not [NoCommunicationWithOdometer](#NoCommunicationWithOdometer)(k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0186], [iTC\_CC-SyAD-1161], [iTC\_CC-SyAD-0960]

[End]

[iTC\_CC\_ATP-SwRS-0636]

WheelMinSpeed，里程计测得车轮最小速度，非负值。

def [WheelMinSpeed](#WheelMinSpeed)(k):

return [round.floor](#roundfloor)([abs](#abs)([WheelMinimumMovement](#WheelMinimumMovement)(k)) / ATP\_CYCLE\_TIME)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136]

[End]

[iTC\_CC\_ATP-SwRS-0204]

WheelMaxSpeed，ATP根据里程计测得位移计算车轮最大速度，该值为非负数，并且向上取整。

ATP calculates the maximum wheel speed according to the maximum wheel movement; this value is non-negative and rounded up.

def [WheelMaxSpeed](#WheelMaxSpeed)(k):

return [round.ceil](#roundceil)([abs](#abs)([WheelMaximumMovement](#WheelMaximumMovement)(k)) / **ATP\_CYCLE\_TIME**)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0146]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [OdometerState](#OdometerState) | √ | √ | ENUM\_ODOMETER\_STATE |
| [ValidWheelKinematic](#ValidWheelKinematic) | √ | √ | BOOLEAN |
| [WheelMaxSpeed](#WheelMaxSpeed) | √ | √ | NUMERIC\_32 |
| [WheelMaximumMovement](#WheelMaximumMovement) | √ | √ | NUMERIC\_32 |
| [WheelMinSpeed](#WheelMinSpeed) | √ | √ | NUMERIC\_32 |
| [WheelMinimumMovement](#WheelMinimumMovement) | √ | √ | NUMERIC\_32 |

## F24-Compensate Sliding Slipping Effect

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [MaxCogCalibration](#MaxCogCalibration) | Internal | F28-Calibrate Wheel Movement |
| [OdometerState](#OdometerState) | Internal | F23-Manage the Odometer State |
| [TeethCounter](#TeethCounter) | Internal | F22-Monitor the Odometer |
| [WheelFilteredStopped](#WheelFilteredStopped) | Internal | F22-Monitor the Odometer |
| [WheelMaximumMovement](#WheelMaximumMovement) | Internal | F23-Manage the Odometer State |
| [WheelMaxSpeed](#WheelMaxSpeed) | Internal | F23-Manage the Odometer State |
| [WheelMinimumMovement](#WheelMinimumMovement) | Internal | F23-Manage the Odometer State |
| [WheelMinSpeed](#WheelMinSpeed) | Internal | F23-Manage the Odometer State |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [AverageWheelAcceleration](#AverageWheelAcceleration) | √ | √ | NUMERIC\_32 |
| [FilteredWheelAcceleration](#FilteredWheelAcceleration) | √ | √ | NUMERIC\_32 |
| [InstantaneousWheelAcceleration](#InstantaneousWheelAcceleration) | √ | √ | NUMERIC\_32 |
| [MaxMotionDuringBrakingOrSliding](#MaxMotionDuringBrakingOrSliding) | √ | √ | NUMERIC\_32 |
| [MaxMotionOdometerSignChanged](#MaxMotionOdometerSignChanged) | √ | √ | BOOLEAN |
| [MotionUnderEstimationState](#MotionUnderEstimationState) | √ | √ | ENUM\_SLIPPING\_STATE |
| [OdometerAxleMotorized](#OdometerAxleMotorized) | √ | √ | BOOLEAN |
| [OverestimatedMotionMax](#OverestimatedMotionMax) | √ | √ | NUMERIC\_32 |
| [OverestimatedMotionMin](#OverestimatedMotionMin) | √ | √ | NUMERIC\_32 |
| [SlidingEnded](#SlidingEnded) | √ | √ | BOOLEAN |
| [SlidingExcess](#SlidingExcess) | √ | √ | BOOLEAN |
| [SlipSlideModellingFault](#SlipSlideModellingFault) | √ | √ | BOOLEAN |
| [SlippingEnded](#SlippingEnded) | √ | √ | BOOLEAN |
| [SlippingExcess](#SlippingExcess) | √ | √ | BOOLEAN |
| [StartBrakingMovementMax](#StartBrakingMovementMax) | √ | √ | NUMERIC\_32 |
| [StartBrakingMovementMin](#StartBrakingMovementMin) | √ | √ | NUMERIC\_32 |
| [StartMotoringMovementMin](#StartMotoringMovementMin) | √ | √ | NUMERIC\_32 |
| [StartSlidingSpeed](#StartSlidingSpeed) | √ | √ | NUMERIC\_32 |
| [StartSlippingSpeed](#StartSlippingSpeed) | √ | √ | NUMERIC\_32 |
| [TimeInSliding](#TimeInSliding) | √ | √ | NUMERIC\_32 |
| [TimeInSlipping](#TimeInSlipping) | √ | √ | NUMERIC\_32 |
| [UnderestimatedMotionMax](#UnderestimatedMotionMax) | √ | √ | NUMERIC\_32 |
| [UnderestimatedMotionMin](#UnderestimatedMotionMin) | √ | √ | NUMERIC\_32 |

### Processing

由于编码里程计安装在列车的制动轴上，因此当列车制动时，钢轨和车轮之间有可能会发生“打滑”现象，导致通过里程计测得的车轮位移小于实际的列车位移（即车轮的转动速度慢于列车实际前进的速度）。为补偿这种情况，ATP软件根据里程计测得的加速度变化，对测得位移进行补偿，得到过估的列车“实际位移”，并据此过估列车的速度，从而保证安全。

As the coded odometer installed on the train brake axle, there is maybe sliding phenomenon between rail and wheel when the train braking. So it will lead to the situation that the detected wheel movement is less than the actual train movement (i.e. the rolling speed of wheel is slower than the actual train speed). In order to avoid this situation, according to the variables of acceleration tested by odometer, ATP will over-estimate the movement and get the over-estimated actual train movement to estimate the train speed for safety consideration.

里程计应尽量避免安装在列车的牵引轴上，若无法避免，则ATP还需对牵引时产生的空转进行补偿。即空转时，里程计测得的车轮转动速度要大于车体实际的移动速度。

[iTC\_CC\_ATP-SwRS-0201]

InstantaneousWheelAcceleration，在进行最大位移过估算法之前，需计算瞬时车轮加速度（为减少采样周期过短使得采样误差导致的加速度大幅变化，ATP使用相邻2个周期的算术平均加速度作为瞬时加速度）。

When wheel motion and acceleration are measurable, then instantaneous acceleration computed according following expression:

def [InstantaneousWheelAcceleration](#InstantaneousWheelAcceleration)(k):

return (([abs](#abs)([TeethCounter](#TeethCounter)(k) - [TeethCounter](#TeethCounter)(k-2))

- [abs](#abs)([TeethCounter](#TeethCounter)(k-2) - [TeethCounter](#TeethCounter)(k-4)))

\* [MaxCogCalibration](#MaxCogCalibration)(k-1) / [pow](#pow)(2\*ATP\_CYCLE\_TIME))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0147], [iTC\_CC\_ATP\_SwHA-0219]

[End]

[iTC\_CC\_ATP-SwRS-0202]

FilteredWheelAcceleration，在进行最大位移过估算法之前，ATP需计算**FILTERED\_ACCELERATION\_NB**个周期的滤波平均加速度

[FilteredWheelAcceleration](#FilteredWheelAcceleration) measurement is the average of [InstantaneousWheelAcceleration](#InstantaneousWheelAcceleration) over **FILTERED\_ACCELERATION\_NB** cycles for filtering the fluctuation causing by the sampling period.

[FilteredWheelAcceleration](#FilteredWheelAcceleration)(k)

= ([InstantaneousWheelAcceleration](#InstantaneousWheelAcceleration)(k)

+ [InstantaneousWheelAcceleration](#InstantaneousWheelAcceleration)(k-1)

+ ...

+ [InstantaneousWheelAcceleration](#InstantaneousWheelAcceleration)(k-**FILTERED\_ACCELERATION\_NB**+1))

/ **FILTERED\_ACCELERATION\_NB**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0147], [iTC\_CC\_ATP\_SwHA-0219]

[End]

[iTC\_CC\_ATP-SwRS-0203]

AverageWheelAcceleration，在进行最大位移过估算法之前，ATP需计算**AVERAGE\_ACCELERATION\_NB**个周期的平均车轮加速度

When wheel motion and acceleration are measurable, sliding average acceleration at cycle k defined by following expression:

[AverageWheelAcceleration](#AverageWheelAcceleration)(k)

= ([InstantaneousWheelAcceleration](#InstantaneousWheelAcceleration)(k)

+ [InstantaneousWheelAcceleration](#InstantaneousWheelAcceleration)(k-1)

+ ...

+ [InstantaneousWheelAcceleration](#InstantaneousWheelAcceleration)(k-**AVERAGE\_ACCELERATION\_NB**+1))

/ **AVERAGE\_ACCELERATION\_NB**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0147], [iTC\_CC\_ATP\_SwHA-0219]

[End]

#### Sliding state management

[iTC\_CC\_ATP-SwRS-0205]

StartBrakingMovementMax，记录由**COASTING**→**BRAKING**，**COASTING**→**SLIDING**，或**BRAKING**→**SLIDING**状态时的最大位移。

ATP records the maximum movement when the state transferring from **COASTING** to **BRAKING** or **SLIDING**, or from **BRAKING** to **SLIDING**.

def [StartBrakingMovementMax](#StartBrakingMovementMax)(k):

if (Initialization

or ([MotionOverEstimationState](#MotionOverEstimationState)(k) is **COASTING**)):

return 0

elif (([MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **COASTING**

and [MotionOverEstimationState](#MotionOverEstimationState)(k) is **BRAKING**)

or ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **COASTING**

and [MotionOverEstimationState](#MotionOverEstimationState)(k) is **SLIDING**)

or ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **BRAKING**

and [MotionOverEstimationState](#MotionOverEstimationState)(k) is **SLIDING**))):

return [MaximumTrainMotion](#MaximumTrainMotion)(k-1)

else:

return [StartBrakingMovementMax](#StartBrakingMovementMax)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0226]

MaxMotionDuringBrakingOrSliding，在制动或者打滑状态下反向运行的最大位移.

ATP shall record the reversed motions during the braking or sliding state.

if ([StartBrakingMovementMax](#StartBrakingMovementMax)(k-1) > 0)

[MaxMotionDuringBrakingOrSliding](#MaxMotionDuringBrakingOrSliding)(k)

= [min](#min)(([MaxMotionDuringBrakingOrSliding](#MaxMotionDuringBrakingOrSliding)(k-1) + [WheelMaximumMovement](#WheelMaximumMovement)(k)), 0)

elif ([StartBrakingMovementMax](#StartBrakingMovementMax)(k-1) < 0)

[MaxMotionDuringBrakingOrSliding](#MaxMotionDuringBrakingOrSliding)(k)

= [max](#max)(([MaxMotionDuringBrakingOrSliding](#MaxMotionDuringBrakingOrSliding)(k-1) + [WheelMaximumMovement](#WheelMaximumMovement)(k)), 0)

else:

[MaxMotionDuringBrakingOrSliding](#MaxMotionDuringBrakingOrSliding) = 0

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1159]

[End]

[iTC\_CC\_ATP-SwRS-0227]

MaxMotionOdometerSignChanged，用于监控是否发生了测得车轮位移反向.

If the reversed motion during braking or sliding state is greater than a project defined distance, ATP shall consider the motion sign changed.

def [MaxMotionOdometerSignChanged](#MaxMotionOdometerSignChanged)(k):

return ([sign](#sign)([StartBrakingMovementMax](#StartBrakingMovementMax)(k-1)) != [sign](#sign)([WheelMaximumMovement](#WheelMaximumMovement)(k))

and ([abs](#abs)([MaxMotionDuringBrakingOrSliding](#MaxMotionDuringBrakingOrSliding)(k))

> [ATPsetting](#ATPsetting).OdoMinDistAfterSenseChange))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1159]

[End]

[iTC\_CC\_ATP-SwRS-0206]

StartSlidingSpeed，记录由**COASTING**或**BRAKING**进入**SLIDING**状态时的速度。

ATP shall record the speed when the train begins to slide.

if (Initialization

or ([MotionOverEstimationState](#MotionOverEstimationState)(k) == **COASTING**)

or ([MotionOverEstimationState](#MotionOverEstimationState)(k) == **BRAKING**))

[StartSlidingSpeed](#StartSlidingSpeed) = 0

elif ((([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **COASTING**)

or ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **BRAKING**))

and ([MotionOverEstimationState](#MotionOverEstimationState)(k) == **SLIDING**))

[StartSlidingSpeed](#StartSlidingSpeed) = [WheelMaxSpeed](#WheelMaxSpeed)(k-1)

else:

[StartSlidingSpeed](#StartSlidingSpeed) = [StartSlidingSpeed](#StartSlidingSpeed)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0207]

TimeInSliding，记录在**SLINDING**状态下持续了多少个周期.

ATP shall record how many cycles staying in **SLIDING** state.

if (Initialization

or (([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **SLIDING**)

and ([MotionOverEstimationState](#MotionOverEstimationState)(k) != **SLIDING**)))

[TimeInSliding](#TimeInSliding) = 0

elif ([MotionOverEstimationState](#MotionOverEstimationState)(k) == **SLIDING**)

[TimeInSliding](#TimeInSliding) = [TimeInSliding](#TimeInSliding)(k-1) + 1

else:

[TimeInSliding](#TimeInSliding) = [TimeInSliding](#TimeInSliding)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0200]

对于车辆位移的打滑空转补偿状态MotionOverEstimationState如下：

* **COASTING**, 无打滑发生；
* **BRAKING**, 一般刹车，进行打滑补偿Kslide；
* **SLIDING**, 可补偿的打滑状态；
* **SKIDDING**, 无法靠里程计补偿的打滑或空转状态。

各个状态的转换关系如Figure 5‑8所示。

ATP software shall use the over-estimation model for train movement provided by Figure 5‑8 state-diagram. The maximum and minimum train motion shall overestimate based on different state as follows:

* **COASTING**. There is not sliding effect during on train coasting or motoring, so ATP need not to overestimate train motion.
* **BRAKING**. When train brakes, which means the measured acceleration is less than the [ATPsetting](#ATPsetting).BrakingStartAcc (normally -0.3m/s^2), ATP shall overestimated the maximum train motion 15% at most.
* **SLIDING**, When the measured acceleration is less than the [ATPsetting](#ATPsetting).SlidingStartAcc (-2.5m/s^2 normally) or the average acceleration is less than [ATPsetting](#ATPsetting).BrakingStartAcc, ATP shall use the train motion before **SLIDING** as the current train motion.
* **SKIDDING**, If train slides or slips excessively, ATP shall consider odometer motion untrustworthy.



Figure 5‑8 Processing of over estimation state

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0131], [iTC\_CC\_ATP\_SwHA-0071]

[End]

[iTC\_CC\_ATP-SwRS-0209]

当满足以下条件时，[MotionOverEstimationState](#MotionOverEstimationState)由**COASTING**转入**BRAKING**，并执行：

The state transfers from “**COASTING**” to “**BRAKING**” when:

if (([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **COASTING**)

and ([WheelFilteredStopped](#WheelFilteredStopped)(k) != **True**)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k)< [ATPsetting](#ATPsetting).BrakingStartAcc)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k)>= [ATPsetting](#ATPsetting).SlidingStartAcc)

and [OdometerState](#OdometerState)(k) is **INITIALIZED**)

[MotionOverEstimationState](#MotionOverEstimationState) = **BRAKING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0072], [iTC\_CC-SyAD-0133]

[End]

[iTC\_CC\_ATP-SwRS-0210]

The [MotionOverEstimationState](#MotionOverEstimationState) transfers from “**COASTING**” to “**SLIDING**” when:

if (([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **COASTING**)

and ([WheelFilteredStopped](#WheelFilteredStopped)(k) != **True**)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) < [ATPsetting](#ATPsetting).SlidingStartAcc)

and [OdometerState](#OdometerState)(k) is **INITIALIZED**)

[MotionOverEstimationState](#MotionOverEstimationState) = **SLIDING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0073]

[End]

[iTC\_CC\_ATP-SwRS-0213]

The [MotionOverEstimationState](#MotionOverEstimationState) transfers from “**BRAKING**” to “**SLIDING**” when:

if ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **BRAKING**

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) < [ATPsetting](#ATPsetting).SlidingStartAcc)

and ([AverageWheelAcceleration](#AverageWheelAcceleration)(k) < [ATPsetting](#ATPsetting).BrakingStartAcc)

and ([OdometerState](#OdometerState)(k) is **INITIALIZED**)

and (not [MaxMotionOdometerSignChanged](#MaxMotionOdometerSignChanged)(k)))

[MotionOverEstimationState](#MotionOverEstimationState) = **SLIDING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0075]

[End]

[iTC\_CC\_ATP-SwRS-0214]

The [MotionOverEstimationState](#MotionOverEstimationState) transfers from “**BRAKING**” to “**COASTING**” when:

if ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **BRAKING**

and (([AverageWheelAcceleration](#AverageWheelAcceleration)(k)>= [ATPsetting](#ATPsetting).BrakingStartAcc)

or ([OdometerState](#OdometerState)(k) is **INVALID**)

or ([MaxMotionOdometerSignChanged](#MaxMotionOdometerSignChanged)(k))))

[MotionOverEstimationState](#MotionOverEstimationState) = **COASTING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0076]

[End]

[iTC\_CC\_ATP-SwRS-0450]

The [MotionOverEstimationState](#MotionOverEstimationState) transfers from “**SLIDING**” to “**COASTING**” when:

if (([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **SLIDING**)

and (([OdometerState](#OdometerState)(k) is **INVALID**)

or ([MaxMotionOdometerSignChanged](#MaxMotionOdometerSignChanged)(k) == **True**))

[MotionOverEstimationState](#MotionOverEstimationState) = **COASTING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0078]

[End]

**NOTES**：

打滑状态下列车位移的补偿方法基于以下假定：

由于判断进入**SLIDING**状态的阈值[ATPsetting](#ATPsetting).SlidingStartAcc为一个较大的减速度（典型值为-2.3m/s^2），在正常制动过程中不可能达到（列车的全常用制动一般略小于-1m/s^2），所以只可能是在紧急制动时，才进入打滑状态。由于紧急制动施加状态下列车牵引被切除，而车辆的紧急制动最小保障率绝对值要大于坡度导致的加速度（由项目保证）。所以，在**SLIDING**过程中，列车的真实速度是逐渐减小的，不可能出现打滑后车速比打滑前还大的情形。然而，如果测得车轮减速度的绝对值过大，或者在打滑状态下过长时间，则认为打滑补偿算法失效。

The principle of overestimation in **SLIDING** state based on the following assumptions:

Because the threshold deceleration (the typical value is -2.3m/s^2) used to detect sliding is far less than the full service braking deceleration (normally -1m/s^2), it is not possible to reach the **SLIDING** state unless the emergency brake applied. The project guaranteed that the absolute value of the minimum emergency brake deceleration is greater than the acceleration due to track gradient, and the rolling stock must cut off the traction during EB applied. Therefore, during **SLIDING** state, the real speed of the train must gradually reduce. However, if the measured absolute value of the wheel deceleration is too large or too long in the sliding conditions, ATP shall consider the overestimation algorithm as failure.

[iTC\_CC\_ATP-SwRS-0451]

SlidingEnded，判断是否结束打滑状态的条件之一。

At cycle k, if motion overestimation status is **SLIDING**, ATP shall consider that sliding effect is ended ([SlidingEnded](#SlidingEnded)) if [FilteredWheelAcceleration](#FilteredWheelAcceleration) is strictly less than [ATPsetting](#ATPsetting).SlippingStopAcc and strictly greater than [ATPsetting](#ATPsetting).SlidingStopAcc for more than [ATPsetting](#ATPsetting).SlidingGripRecoveryTime.

if ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **SLIDING**

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) < [ATPsetting](#ATPsetting).SlippingStopAcc)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) > [ATPsetting](#ATPsetting).SlidingStopAcc)

and ([MotionOverEstimationState](#MotionOverEstimationState)(k-2) == **SLIDING**)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k-1) < [ATPsetting](#ATPsetting).SlippingStopAcc)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k-1) > [ATPsetting](#ATPsetting).SlidingStopAcc)

...

and ([MotionOverEstimationState](#MotionOverEstimationState)(k-[ATPsetting](#ATPsetting).SlidingGripRecoveryTime) == **SLIDING**)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k-[ATPsetting](#ATPsetting).SlidingGripRecoveryTime+1)

< [ATPsetting](#ATPsetting).SlippingStopAcc)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k-[ATPsetting](#ATPsetting).SlidingGripRecoveryTime+1)

> [ATPsetting](#ATPsetting).SlidingStopAcc))

[SlidingEnded](#SlidingEnded) = **True**

else:

[SlidingEnded](#SlidingEnded) = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0079]

[End]

[iTC\_CC\_ATP-SwRS-0218]

The [MotionOverEstimationState](#MotionOverEstimationState) transfers from “**SLIDING**” to “**BRAKING**” when:

if (([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **SLIDING**)

and ([OdometerState](#OdometerState)(k) is **INITIALIZED**)

and (not [MaxMotionOdometerSignChanged](#MaxMotionOdometerSignChanged)(k))

and ([TimeInSliding](#TimeInSliding)(k-1) <= [ATPsetting](#ATPsetting).SlidingTimeout)

and (0 < (|[StartSlidingSpeed](#StartSlidingSpeed)(k-1)| + [TimeInSliding](#TimeInSliding)(k-1)\* [ATPsetting](#ATPsetting).SlidingStopAcc))

and ((|[StartSlidingSpeed](#StartSlidingSpeed)(k-1)| + [TimeInSliding](#TimeInSliding)(k-1)\* [ATPsetting](#ATPsetting).SlidingStopAcc)

< |[WheelMaxSpeed](#WheelMaxSpeed)(k)|)

and ([SlidingEnded](#SlidingEnded)(k) == **True**))

[MotionOverEstimationState](#MotionOverEstimationState) = **BRAKING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0079]

[End]

[iTC\_CC\_ATP-SwRS-0452]

SlidingExcess，测得的加速度在项目配置范围内满足一定时间，是ATP判断过度打滑的必要条件之一。

At cycle k, if motion overestimation status is **SLIDING**, ATP shall consider that sliding is excess ([SlidingExcess](#SlidingExcess)) if [FilteredWheelAcceleration](#FilteredWheelAcceleration) is strictly less than [ATPsetting](#ATPsetting).SlippingStopAcc and strictly greater than [ATPsetting](#ATPsetting).SlidingStopAcc for more than [ATPsetting](#ATPsetting).SlidingExcessTime.

if ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) == **SLIDING**

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) < [ATPsetting](#ATPsetting).SlippingStopAcc)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) > [ATPsetting](#ATPsetting).SlidingStopAcc)

and ([MotionOverEstimationState](#MotionOverEstimationState)(k-2) == **SLIDING**)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k-1) < [ATPsetting](#ATPsetting).SlippingStopAcc)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k-1) > [ATPsetting](#ATPsetting).SlidingStopAcc)

...

and ([MotionOverEstimationState](#MotionOverEstimationState)(k-[ATPsetting](#ATPsetting).SlidingExcessTime) == **SLIDING**)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k-[ATPsetting](#ATPsetting).SlidingExcessTime+1)

< [ATPsetting](#ATPsetting).SlippingStopAcc)

and ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k-[ATPsetting](#ATPsetting).SlidingExcessTime+1)

> [ATPsetting](#ATPsetting).SlidingStopAcc))

[SlidingExcess](#SlidingExcess) = **True**

else:

[SlidingExcess](#SlidingExcess) = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0080]

[End]

[iTC\_CC\_ATP-SwRS-0217]

The [MotionOverEstimationState](#MotionOverEstimationState) transfers from **SLIDING** to **SKIDDING** when:

if ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **SLIDING**

and [OdometerState](#OdometerState)(k) is **INITIALIZED**

and not [MaxMotionOdometerSignChanged](#MaxMotionOdometerSignChanged)(k)

and ([TimeInSliding](#TimeInSliding)(k-1) > [ATPsetting](#ATPsetting).SlidingTimeout

or ([StartSlidingSpeed](#StartSlidingSpeed)(k-1)+ [TimeInSliding](#TimeInSliding)(k-1) \* [ATPsetting](#ATPsetting).SlidingStopAcc) <= **0**

or ((([StartSlidingSpeed](#StartSlidingSpeed)(k-1) + [TimeInSliding](#TimeInSliding)(k-1)\* [ATPsetting](#ATPsetting).SlidingStopAcc)

>= [WheelMaxSpeed](#WheelMaxSpeed)(k))

and [SlidingExcess](#SlidingExcess)(k))))

[MotionOverEstimationState](#MotionOverEstimationState) = **SKIDDING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0081], [iTC\_CC\_ATP\_SwHA-0082]

[End]

**NOTES**：

由于当前项目应用的车辆都装有ABS防抱死系统，使得在制动时列车的加速度不会连续若干周期小于[ATPsetting](#ATPsetting).SlidingStopAcc。因此，如果某周期瞬间加速度小于[ATPsetting](#ATPsetting).SlidingStopAcc；并且在之后的测得加速度满足[SlidingExcess](#SlidingExcess)条件，那么，下列两个条件可以同时成立：

Because the application of anti-lock braking system for the train of current project, makes the brake acceleration cannot continuous less than [ATPsetting](#ATPsetting).SlidingStopAcc for serious cycles. Therefore, if there was an unexpected instantaneous acceleration less than [ATPsetting](#ATPsetting).SlidingStopAcc, and the after cycles' acceleration met the criteria of [SlidingExcess](#SlidingExcess), then the following two conditions can hold simultaneously.

|[StartSlidingSpeed](#StartSlidingSpeed)(k-1)| + [TimeInSliding](#TimeInSliding)(k) \* [ATPsetting](#ATPsetting).SlidingStopAcc >= |[WheelMaxSpeed](#WheelMaxSpeed)(k)|)

AND ([SlidingExcess](#SlidingExcess)(k) == **True**)

[iTC\_CC\_ATP-SwRS-0220]

The [MotionOverEstimationState](#MotionOverEstimationState) transfers from **SKIDDING** to **COASTING** when:

if ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **SKIDDING**

and ([WheelFilteredStopped](#WheelFilteredStopped)(k)

or [OdometerState](#OdometerState)(k) is **INVALID**))

[MotionOverEstimationState](#MotionOverEstimationState) = **COASTING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0084]

[End]

[iTC\_CC\_ATP-SwRS-0735]

StartBrakingMovementMin，记录由**COASTING**进入**BRAKING**，**COASTING**进入**SLIDING**，或者**BRAKING**进入**SLIDING**状态时的最小位移。

ATP records the minimum movement when the state transferring from **COASTING** to **BRAKING** or **SLIDING**, or from **BRAKING** to **SLIDING**.

def [StartBrakingMovementMin](#StartBrakingMovementMin)(k):

if (Initialization

or [OdometerState](#OdometerState)(k-1) is not **INITIALIZED**

or ([MotionOverEstimationState](#MotionOverEstimationState)(k) is **COASTING**)):

return 0

elif (([MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **COASTING**

and [MotionOverEstimationState](#MotionOverEstimationState)(k) is **BRAKING**)

or ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **COASTING**

and [MotionOverEstimationState](#MotionOverEstimationState)(k) is **SLIDING**)

or ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **BRAKING**

and [MotionOverEstimationState](#MotionOverEstimationState)(k) is **SLIDING**)):

return [MinimumTrainMotion](#MinimumTrainMotion)(k-1)

else:

return [StartBrakingMovementMin](#StartBrakingMovementMin)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0787]

OverestimatedMotionMin，根据打滑状态机，对里程计测得的最小位移进行补偿。

* 在**BRAKING**或**SLIDING**状态时，由于车辆ABS的作用，会在瞬间释放制动而使得转速突然增大，可能接近但不会大于进入制动状态时刻的速度。而由于获取里程计读值有1个齿的采样误差，在该误差的作用下，可能会使得测得位移大于进入制动状态时刻的位移，即出现测得车轮最小位移大于列车最大位移的情形。为防止这种情况，需要对列车最小位移进行调整，即始终使用进入制动状态时刻与测得车轮最小位移中绝对值较小的一个。
* 其他情况，无需补偿，使用测得位移。

def [OverstimatedMotionMin](#OverstimatedMotionMin)(k):

if (sign([StartBrakingMovementMin](#StartBrakingMovementMin)(k)) == sign([WheelMinimumMovement](#WheelMinimumMovement)(k))

and ([MotionOverEstimationState](#MotionOverEstimationState) (k) is **BRAKING**

or [MotionOverEstimationState](#MotionOverEstimationState) (k) is **SLIDING**)):

if ([StartBrakingMovementMin](#StartBrakingMovementMin)(k) >= 0):

return min([StartBrakingMovementMin](#StartBrakingMovementMin)(k), [WheelMinimumMovement](#WheelMinimumMovement)(k))

else:

return (-1 \* min(abs([StartBrakingMovementMin](#StartBrakingMovementMin)(k)), abs([WheelMinimumMovement](#WheelMinimumMovement)(k))))

else:

return [WheelMinimumMovement](#WheelMinimumMovement)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0131], [iTC\_CC\_ATP\_SwHA-0070]

[End]

[iTC\_CC\_ATP-SwRS-0788]

OverestimatedMotionMax，根据打滑状态机，对里程计测得的最大位移进行补偿。

如果本周期在**BRAKING**状态，按如下规则更新列车最大位移：

* 如果本周期测得车轮位移与[StartBrakingMovementMax](#StartBrakingMovementMax)方向相同，且前者的绝对值大于后者的绝对值，表明由于采样齿数波动，测得位移大于[StartBrakingMovementMax](#StartBrakingMovementMax)，此时使用测得位移作为最大列车位移；
* 否则，根据配置对测得位移进行补偿，取[StartBrakingMovementMax](#StartBrakingMovementMax)与补偿后的测得位移中绝对值较小的一个，位移方向与[StartBrakingMovementMax](#StartBrakingMovementMax)相同。

In **BRAKING** state, the maximum train motion overestimated as [ATPsetting](#ATPsetting).SlidingCoefficient (15% normally) at most. If the overestimated motion has greater than the start braking movement, ATP shall use the start breaking movement as the current train motion. That said the train speed during braking could not faster than before.

如果本周期在**SLIDING**状态时，按如下规则更新列车最大位移：

* 如果本周期测得车轮位移与[StartBrakingMovementMax](#StartBrakingMovementMax)方向相同，且前者的绝对值大于后者的绝对值，表明由于采样齿数波动，使得测得位移大于[StartBrakingMovementMax](#StartBrakingMovementMax)。此时使用测得位移作为最大列车位移；
* 否则，使用[StartBrakingMovementMax](#StartBrakingMovementMax)

In state **SLIDING**:

* If both [WheelMaximumMovement](#WheelMaximumMovement) and [StartBrakingMovementMax](#StartBrakingMovementMax) are same direction, and the absolute value of the former is greater than the absolute value of the latter, indicating that due to the sampling error makes the measured movement greater than [StartBrakingMovementMax](#StartBrakingMovementMax). In this case, ATP shall uses [WheelMaximumMovement](#WheelMaximumMovement) as current train maximum motion.
* Otherwise, uses [StartBrakingMovementMax](#StartBrakingMovementMax) as train maximum motion.

其他状态下，无需对测得最大位移进行补偿。

In other state (**COASTING**, **SKIDDING**), uses measured wheel maximum movement as current overestimated maximum train motion.

def [OverestimatedMotionMax](#OverestimatedMotionMax)(k):

if [MotionOverEstimationState](#MotionOverEstimationState)(k) is **BRAKING**:

if (sign([StartBrakingMovementMax](#StartBrakingMovementMax)(k)) == sign([WheelMaximumMovement](#WheelMaximumMovement)(k))

and abs([WheelMaximumMovement](#WheelMaximumMovement)(k)) > abs([StartBrakingMovementMax](#StartBrakingMovementMax)(k))):

return [WheelMaximumMovement](#WheelMaximumMovement)(k)

elif ([StartBrakingMovementMax](#StartBrakingMovementMax)(k) >= **0**):

return min(abs([StartBrakingMovementMax](#StartBrakingMovementMax)(k)),

abs([WheelMaximumMovement](#WheelMaximumMovement)(k) \* ATPsetting.SlidingCoefficient))

else:

return **-1** \* min(abs([StartBrakingMovementMax](#StartBrakingMovementMax)(k)),

abs([WheelMaximumMovement](#WheelMaximumMovement)(k) \* ATPsetting.SlidingCoefficient))

elif [MotionOverEstimationState](#MotionOverEstimationState)(k) is **SLIDING**:

if (sign([StartBrakingMovementMax](#StartBrakingMovementMax)(k)) == sign([WheelMaximumMovement](#WheelMaximumMovement)(k))

and abs([WheelMaximumMovement](#WheelMaximumMovement)(k)) > abs([StartBrakingMovementMax](#StartBrakingMovementMax)(k))):

return [WheelMaximumMovement](#WheelMaximumMovement)(k)

else:

return [StartBrakingMovementMax](#StartBrakingMovementMax)(k)

else: # Coasting and Skidding

return [WheelMaximumMovement](#WheelMaximumMovement)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0074]

[End]

#### Slipping state management

[iTC\_CC\_ATP-SwRS-0769]

StartSlippingSpeed，记录由**COASTING**或**MOTORING**进入**SLIPPING**状态时的速度。

ATP shall record the speed when the train begins to slip.

def [StartSlippingSpeed](#StartSlippingSpeed)(k):

if (Initialization

or [OdometerState](#OdometerState)(k-1) is not **INITIALIZED**

or [MotionUnderEstimationState](#MotionUnderEstimationState)(k) is **COASTING**

or [MotionUnderEstimationState](#MotionUnderEstimationState)(k) is **MOTORING**):

return **0**

elif (([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **COASTING**

or [MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **MOTORING**)

and [MotionUnderEstimationState](#MotionUnderEstimationState)(k) is **SLIPPING**):

return [WheelMinSpeed](#WheelMinSpeed)(k-1)

else:

return [StartSlippingSpeed](#StartSlippingSpeed)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0131]

[End]

[iTC\_CC\_ATP-SwRS-0770]

TimeInSlipping，记录在**SLIPPING**状态下持续了多少个周期.

ATP shall record how many cycles staying in **SLIPPING** state.

def [TimeInSlipping](#TimeInSlipping)(k):

if (Initialization

or ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **SLIPPING**

and [MotionUnderEstimationState](#MotionUnderEstimationState)(k) is not **SLIPPING**)):

return 0

elif ([MotionUnderEstimationState](#MotionUnderEstimationState)(k) is **SLIPPING**):

return [TimeInSlipping](#TimeInSlipping)(k-1) + 1

else:

return [TimeInSlipping](#TimeInSlipping)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0793]

OdometerAxleMotorized，表示需考虑里程计所安在车轴牵引导致的空转。

If the project that odometer installed on the traction axle of the train, ATP shall consider the slipping effect to impact the underestimation of measured wheel movement.

def [OdometerAxleMotorized](#OdometerAxleMotorized)(k):

return not [ATPsetting](#ATPsetting).OdoNotOnMotorizedAxle

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0131], [iTC\_CC\_ATP\_SwHA-0070]

[End]

[iTC\_CC\_ATP-SwRS-0199]

对于车辆位移的打滑空转补偿状态MotionUnderEstimationState如下：

* **COASTING**, 无打滑发生；
* **MOTORING**，一般牵引状态（仅在需考虑空转补偿的项目）；
* **SLIPPING**，可补偿的空转状态（仅在需考虑空转补偿的项目）；
* **SKIDDING**, 无法靠里程计补偿的打滑或空转状态。

各个状态的转换关系如Figure 5‑9所示。

ATP software shall use the over-estimation model for train movement provided by Figure 5‑9 state-diagram. The maximum and minimum train motion shall overestimate based on different state as follows:

* **COASTING**. There is not sliding effect during on train coasting or motoring, so ATP need not to overestimate train motion.
* **MOTORING**, normal traction state (only consdering odometer installed on motorized axle).
* **SLIPPING**, wheel slipping happen (only consdering odometer installed on motorized axle).
* **SKIDDING**, If train slides or slips excessively, ATP shall consider odometer motion untrustworthy.



Figure 5‑9 Processing of under estimation state

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0131], [iTC\_CC\_ATP\_SwHA-0071]

[End]

[iTC\_CC\_ATP-SwRS-0771]

The [MotionUnderEstimationState](#MotionUnderEstimationState) transfers from “**COASTING**” to “**MOTORING**” when:

if ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **COASTING**

and [OdometerAxleMotorized](#OdometerAxleMotorized)(k)

and [FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) <= ATPsetting.SlippingStartAcc

and [FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) > ATPsetting.TractionStartAcc

and [OdometerState](#OdometerState)(k) is **INITIALIZED**)

[MotionUnderEstimationState](#MotionUnderEstimationState) = **MOTORING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC-SyAD-0133]

[End]

[iTC\_CC\_ATP-SwRS-0772]

The [MotionUnderEstimationState](#MotionUnderEstimationState) transfers from “**COASTING**” to “**SLIPPING**” when:

if ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **COASTING**

and [OdometerAxleMotorized](#OdometerAxleMotorized)(k)

and [FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) > [ATPsetting](#ATPsetting).SlippingStartAcc

and [OdometerState](#OdometerState)(k) is **INITIALIZED**)

[MotionUnderEstimationState](#MotionUnderEstimationState) = **SLIPPING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0073]

[End]

[iTC\_CC\_ATP-SwRS-0773]

The [MotionUnderEstimationState](#MotionUnderEstimationState) transfers from “**MOTORING**” to “**SLIPPING**” when:

if ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **MOTORING**

and [FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) > [ATPsetting](#ATPsetting).SlippingStartAcc

and [AverageWheelAcceleration](#AverageWheelAcceleration)(k) > [ATPsetting](#ATPsetting).MotoringStartAc)

and [OdometerState](#OdometerState)(k) is **INITIALIZED**

and [OdometerAxleMotorized](#OdometerAxleMotorized)(k))

[MotionUnderEstimationState](#MotionUnderEstimationState) = **SLIDING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0774]

The [MotionUnderEstimationState](#MotionUnderEstimationState) transfers from **MOTORING** to **COASTING**” when:

if ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **MOTORING**

and ([AverageWheelAcceleration](#AverageWheelAcceleration)(k) <= [ATPsetting](#ATPsetting).TractionStartAcc

or [OdometerState](#OdometerState)(k) is **INVALID**

or not [OdometerAxleMotorized](#OdometerAxleMotorized)(k)))

[MotionUnderEstimationState](#MotionUnderEstimationState) = **COASTING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0775]

The [MotionUnderEstimationState](#MotionUnderEstimationState) transfers from **SLIPPING** to **COASTING** when:

if ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **SLIPPING**

and ([OdometerState](#OdometerState)(k) is **INVALID**

or not [OdometerAxleMotorized](#OdometerAxleMotorized)(k)))

[MotionUnderEstimationState](#MotionUnderEstimationState) = **COASTING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0078]

[End]

[iTC\_CC\_ATP-SwRS-0776]

SlippingEnded，判断是否结束空转状态的条件之一。

def [SlippingEnded](#SlippingEnded)(k):

if ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is not **SLIPPING**):

slipping\_ended\_counter = **0**

return **False**

elif ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) < ATPsetting.SlippingStopAcc

and [FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) > ATPsetting.SlidingStopAcc):

slipping\_ended\_counter = slipping\_ended\_counter **+ 1**

return (slipping\_ended\_counter >= ATPsetting.SlippingGripRecoveryTime)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0777]

The [MotionUnderEstimationState](#MotionUnderEstimationState) transfers from **SLIPPING** to **MOTORING** when:

if ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **SLIPPING**

and [OdometerState](#OdometerState)(k) is **INITIALIZED**

and [OdometerAxleMotorized](#OdometerAxleMotorized)(k)

and [TimeInSlipping](#TimeInSlipping)(k-1) <= [ATPsetting](#ATPsetting).SlippingTimeout

and (([StartSlippingSpeed](#StartSlippingSpeed)(k-1) + [TimeInSlipping](#TimeInSlipping)(k-1)\* [ATPsetting](#ATPsetting).SlippingStopAcc)

> [WheelMinSpeed](#WheelMinSpeed)(k))

and [SlippingEnded](#SlippingEnded)(k))

[MotionUnderEstimationState](#MotionUnderEstimationState) = **MOTORING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0778]

SlippingExcess，测得的加速度在项目配置范围内满足一定时间，是ATP判断过度空转的必要条件之一。

def [SlippingExcess](#SlippingExcess)(k):

if ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is not **SLIPPING**):

slipping\_excess\_counter = **0**

return **False**

elif ([FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) < ATPsetting.SlippingStopAcc

and [FilteredWheelAcceleration](#FilteredWheelAcceleration)(k) > ATPsetting.SlidingStopAcc):

slipping\_excess\_counter = slipping\_excess\_counter + **1**

return (slipping\_excess\_counter >= ATPsetting.SlippingExcessTime)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0779]

The [MotionUnderEstimationState](#MotionUnderEstimationState) transfers from **SLIPPING** to **SKIDDING** when:

if ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **SLIPPING**

and [OdometerState](#OdometerState)(k) is **INITIALIZED**

and [OdometerAxleMotorized](#OdometerAxleMotorized)(k)

and ([TimeInSlipping](#TimeInSlipping)(k-1) > [ATPsetting](#ATPsetting).SlippingTimeout

or ((([StartSlippingSpeed](#StartSlippingSpeed)(k-1) + [TimeInSlipping](#TimeInSlipping)(k-1)\* [ATPsetting](#ATPsetting).SlippingStopAcc)

<= [WheelMinSpeed](#WheelMinSpeed)(k))

and [SlippingExcess](#SlippingExcess)(k))))

[MotionUnderEstimationState](#MotionUnderEstimationState) = **SKIDDING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0794]

The [MotionUnderEstimationState](#MotionUnderEstimationState) transfers from **SKIDDING** to **COASTING** when:

if ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **SKIDDING**

and ([WheelFilteredStopped](#WheelFilteredStopped)(k)

or [OdometerState](#OdometerState)(k) is **INVALID**))

[MotionUnderEstimationState](#MotionUnderEstimationState) = **COASTING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0084]

[End]

[iTC\_CC\_ATP-SwRS-0780]

StartMotoringMovementMin，记录由**COASTING**进入**MOTORING**，**COASTING**进入**SLIPPING**，或者**MOTORING**进入**SLIPPING**状态时的最小位移。

ATP records the minimum movement when the state transferring from **COASTING** to **MOTORING** or **SLIPPING**, or from **MOTORING** to **SLIPPING**.

def [StartMotoringMovementMin](#StartMotoringMovementMin)(k):

if (Initialization

or [OdometerState](#OdometerState)(k-1) is not **INITIALIZED**

or ([MotionUnderEstimationState](#MotionUnderEstimationState)(k) is **COASTING**)):

return 0

elif (([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **COASTING**

and [MotionUnderEstimationState](#MotionUnderEstimationState) (k) is **MOTORING**)

or ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **COASTING**

and [MotionUnderEstimationState](#MotionUnderEstimationState) (k) is **SLIPPING**)

or ([MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is **MOTORING**

and [MotionUnderEstimationState](#MotionUnderEstimationState)(k) is **SLIPPING**)):

return [MinimumTrainMotion](#MinimumTrainMotion)(k-1)

else:

return [StartMotoringMovementMin](#StartMotoringMovementMin)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0795]

UnderestimatedMotionMin，根据空转状态机，对里程计测得的最小位移进行补偿。

* 在**MOTORING**状态下，使用牵引入口位移和将测得位移低估15%补偿后二者较大的一个，作为补偿后的位移。
* 在**SLIPPING**状态下，使用牵引入口位移作为补偿后的位移。

def [UnderestimatedMotionMin](#UnderestimatedMotionMin)(k):

if [MotionUnderEstimationState](#MotionUnderEstimationState)(k) is **MOTORING**:

if ([WheelMinimumMovement](#WheelMinimumMovement)(k) >= 0):

return max(abs([StartMotoringMovementMin](#StartMotoringMovementMin)(k)),

abs([WheelMinimumMovement](#WheelMinimumMovement)(k) \* ATPsetting.SlippingCoefficient))

else:

return -1 \* max(abs([StartMotoringMovementMin](#StartMotoringMovementMin)(k)),

abs([WheelMinimumMovement](#WheelMinimumMovement)(k) \* ATPsetting.SlippingCoefficient))

elif [MotionUnderEstimationState](#MotionUnderEstimationState)(k) is **SLIPPING**:

return [StartMotoringMovementMin](#StartMotoringMovementMin)(k)

else:

return [WheelMinimumMovement](#WheelMinimumMovement)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0796]

UnderestimatedMotionMax，根据空转状态机，对里程计测得的最大位移进行补偿。

def [UnderestimatedMotionMax](#UnderestimatedMotionMax)(k):

return [WheelMaximumMovement](#WheelMaximumMovement)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148]

[End]

#### Slide slip modelling validity

[iTC\_CC\_ATP-SwRS-0191]

SlipSlideDetected，是否检测到打滑空转

For calibration validation purpose, ATP shall consider that slip/side detected if:

* motion overestimation modeling status is not coasting nor braking,
* or motion underestimation modeling status is not coasting nor motoring.

[SlipSlideDetected](#SlipSlideDetected)(k)

= (([MotionOverEstimationState](#MotionOverEstimationState)(k) != **COASTING**

and [MotionOverEstimationState](#MotionOverEstimationState)(k) != **BRAKING**)

or ([MotionUnderEstimationState](#MotionUnderEstimationState)(k) != **COASTING**

and [MotionUnderEstimationState](#MotionUnderEstimationState)(k) != **MOTORING**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0145], [iTC\_CC-SyAD-0148], [iTC\_CC\_ATP\_SwHA-0088]

[End]

[iTC\_CC\_ATP-SwRS-0228]

SlipSlideModellingFault，打滑补偿模型错误

When the overestimation or underesimation state is **SKIDDING**, or the motion signed changed in **BRAKING** or **SLIDING** state, ATP shall consider the overestimation model as fault.

def [SlipSlideModellingFault](#SlipSlideModellingFault)(k):

if ([MotionOverEstimationState](#MotionOverEstimationState)(k) is **SKIDDING**

or [MotionUnderEstimationState](#MotionUnderEstimationState)(k) is **SKIDDING**

or (([MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **BRAKING**

or [MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **SLIDING**)

and [MaxMotionOdometerSignChanged](#MaxMotionOdometerSignChanged)(k)))):

return **True**

elif ([MotionOverEstimationState](#MotionOverEstimationState)(k-1) is **COASTING**

and [MotionUnderEstimationState](#MotionUnderEstimationState)(k-1) is COASTING)

return **False**

else:

return [SlipSlideModellingFault](#SlipSlideModellingFault)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC-SyAD-0192], [iTC\_CC-SyAD-1159], [iTC\_CC\_ATP\_SwHA-0089]

[End]

[iTC\_CC\_ATP-SwRS-0229]

ValidSlipSlideModelling，打滑补偿模型有效

If overestimation model was fault, then ATP considers the model invalid.

def [ValidSlipSlideModelling](#ValidSlipSlideModelling)(k):

if ([ValidSlipSlideModelling](#ValidSlipSlideModelling)(k-1))

return not [SlipSlideModellingFault](#SlipSlideModellingFault)(k)

elif (([MotionOverEstimationState](#MotionOverEstimationState)(k) is **COASTING**)

and [MotionUnderEstimationState](#MotionUnderEstimationState)(k) is **COASTING**

and [WheelFilteredStopped](#WheelFilteredStopped)(k))

return **True**

else:

return [ValidSlipSlideModelling](#ValidSlipSlideModelling)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148], [iTC\_CC-SyAD-0192], [iTC\_CC-SyAD-1159], [iTC\_CC\_ATP\_SwHA-0090]

[End]

[iTC\_CC\_ATP-SwRS-0797]

MaximumSScompensatedMotion，经过打滑空转补偿后的最大位移

def [MaximumSScompensatedMotion](#MaximumSScompensatedMotion)(k):

if [ValidSlipSlideModelling](#ValidSlipSlideModelling)(k):

return (sign([OverestimatedMotionMax](#OverestimatedMotionMax)(k))

\* max(abs([OverestimatedMotionMax](#OverestimatedMotionMax)(k)), abs([UnderestimatedMotionMax](#UnderestimatedMotionMax)(k)))

else:

return [WheelMaximumMovement](#WheelMaximumMovement)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148]

[End]

[iTC\_CC\_ATP-SwRS-0798]

MinimumSScompensatedMotion，经过打滑空转补偿后的最小位移

def [MinimumSScompensatedMotion](#MinimumSScompensatedMotion)(k):

if [ValidSlipSlideModelling](#ValidSlipSlideModelling)(k):

return (sign([OverestimatedMotionMin](#OverestimatedMotionMin)(k))

\* min(abs([OverestimatedMotionMin](#OverestimatedMotionMin)(k)), abs([UnderestimatedMotionMin](#UnderestimatedMotionMin)(k)))

else:

return [WheelMinimumMovement](#WheelMinimumMovement)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0148]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [MaximumSScompensatedMotion](#MaximumSScompensatedMotion) | √ | √ | NUMERIC\_32 |
| [MinimumSScompensatedMotion](#MinimumSScompensatedMotion) | √ | √ | NUMERIC\_32 |
| [MotionOverEstimationState](#MotionOverEstimationState) | √ | √ | ENUM\_SLIDING\_STATE |
| [SlipSlideDetected](#SlipSlideDetected) | √ | √ | BOOLEAN |
| [ValidSlipSlideModelling](#ValidSlipSlideModelling) | √ | √ | BOOLEAN |

## F25-Calculate Radar Speed

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [CBKWrite](#CBKWrite) | External | Interface with VPB |
| [MaximumSScompensatedMotion](#MaximumSScompensatedMotion) | Internal | F24-Compensate Sliding Slipping Effect |
| [MinimumSScompensatedMotion](#MinimumSScompensatedMotion) | Internal | F24-Compensate Sliding Slipping Effect |
| [ValidSlipSlideModelling](#ValidSlipSlideModelling) | Internal | F24-Compensate Sliding Slipping Effect |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [RadarDirection](#RadarDirection) | √ | √ | NUMERIC\_32 |
| [RadarInfo](#RadarInfo) | √ | √ | ST\_RADAR\_INFO |

### Processing

对于配置有多普勒雷达的项目，ATP软件每周期需获取雷达的测速信息，按照配置参数过估后，作为计算列车速度的依据。

[iTC\_CC\_ATP-SwRS-0781]

RadarInfo，ATP软件根据来自VPB板的雷达寄存器，计算雷达信息。

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_RADAR\_INFO | |  |  |
|  | CbkMovement | NUMERIC\_32 | 根据雷达脉冲信息计算的周期位移 |
|  | CbkSpeed | NUMERIC\_32 | 根据雷达脉冲信息计算的周期速度 |
|  | DrsDistance | NUMERIC\_32 | 根据雷达双通道位移信息累计的运行距离 |
|  | DrsMovement | NUMERIC\_32 | 根据雷达双通道位移信息计算的周期位移 |
|  | DrsSpeed | NUMERIC\_32 | 根据雷达速度信息计算的周期速度 |
|  | DrsValid | BOOLEAN | 雷达信息有效（TBD，应根据上述信息相互校验） |
|  | DrsDirection | NUMERIC\_32 | 向前为1，向后为0 |

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=

[End]

[iTC\_CC\_ATP-SwRS-0782]

RadarRawSpeed，直接获取的雷达测速值，该值始终为正。

def [RadarRawSpeed](#RadarRawSpeed)(k):

if [RadarInfo](#RadarInfo)(k).DrsValid:

return [RadarInfo](#RadarInfo)(k).DrsSpeed

else:

return **MAX\_RADAR\_SPEED**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=

[End]

[iTC\_CC\_ATP-SwRS-0783]

RadarDirection，雷达测得方向，向End1方向为+1，向End2为-1，其余为0

def [RadarDirection](#RadarDirection)(k):

if not [RadarInfo](#RadarInfo)(k).DrsValid:

return 0

elif (([CoreId](#CoreId)(k) is **END\_1**

and [RadarInfo](#RadarInfo)(k).DrsDirection > 0)

or ([CoreId](#CoreId)(k) is **END\_2**

and [RadarInfo](#RadarInfo)(k).DrsDirection <= 0)):

return 1

else:

return -1

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=

[End]

ATP应根据项目配置信息对雷达测得的速度进行过估，方式如下：

* 当原始测得速度大于配置的阈值[ATPsetting](#ATPsetting).RadarSpeedThreshold时，最大最小位移将分别按照百分比[ATPsetting](#ATPsetting).RadarDeviationAboveThreshold过估；
* 当原始测得速度小于配置的阈值[ATPsetting](#ATPsetting).RadarSpeedThreshold时，最大最小位移将按照固定速度值[ATPsetting](#ATPsetting).RadarDeviationBelowThreshold过估，速度最小为0。

[iTC\_CC\_ATP-SwRS-0784]

RadarMotionMax，绝对值向上过估的雷达最大位移，向END1方向该值为正，向END2方向该值为负。

def [RadarMotionMax](#RadarMotionMax)(k):

if not [RadarInfo](#RadarInfo)(k).DrsValid:

return 0

elif [RadarRawSpeed](#RadarRawSpeed)(k) >= [ATPsetting](#ATPsetting).RadarSpeedThreshold:

return (**ATP\_CYCLE\_TIME** \* [RadarDirection](#RadarDirection)(k) \* ([RadarRawSpeed](#RadarRawSpeed)(k)

+ [RadarRawSpeed](#RadarRawSpeed)(k) \* [ATPsetting](#ATPsetting).RadarDeviationAboveThreshold / **1000**))

else:

return (**ATP\_CYCLE\_TIME** \* [RadarDirection](#RadarDirection)(k)

\* ([RadarRawSpeed](#RadarRawSpeed)(k) + [ATPsetting](#ATPsetting).RadarDeviationBelowThreshold))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=

[End]

[iTC\_CC\_ATP-SwRS-0785]

RadarMotionMin，绝对值向下过估的雷达最小位移，向END1方向该值为正，向END2方向该值为负。

def [RadarMotionMin](#RadarMotionMin)(k):

if not [RadarInfo](#RadarInfo)(k).DrsValid:

return 0

elif [RadarRawSpeed](#RadarRawSpeed)(k) >= [ATPsetting](#ATPsetting).RadarSpeedThreshold:

return (**ATP\_CYCLE\_TIME** \* [RadarDirection](#RadarDirection)(k) \* ([RadarRawSpeed](#RadarRawSpeed)(k)

- [RadarRawSpeed](#RadarRawSpeed)(k) \* [ATPsetting](#ATPsetting).RadarDeviationAboveThreshold / **1000**))

else:

return (**ATP\_CYCLE\_TIME** \* [RadarDirection](#RadarDirection)(k)

\* max(**0**, [RadarRawSpeed](#RadarRawSpeed)(k) - [ATPsetting](#ATPsetting).RadarDeviationBelowThreshold))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=

[End]

**NOTES**：

仅当项目配置了雷达，且雷达最大最小位移范围与里程计经打滑空转补偿后的最大最小位移范围有交集时，才认为雷达速度有效。现场数据表明，列车持续减速，当里程计读值变为0后，雷达会延迟超过1秒，测速结果才会变为0，此时不能认为雷达测速无效。因此，只有当雷达测得方向与里程计经打滑补偿后的测得方向相反时，才认为雷达无效。

[iTC\_CC\_ATP-SwRS-0786]

RadarSpeedValid，判断雷达速度是否可用

def [RadarSpeedValid](#RadarSpeedValid)(k):

return ([ATPsetting](#ATPsetting).RadarApplied

and [RadarInfo](#RadarInfo)(k).DrsValid

and (not [ValidSlipSlideModelling](#ValidSlipSlideModelling)(k)

or ([RadarDirection](#RadarDirection)(k) \* [MaximumSScompensatedMotion](#MaximumSScompensatedMotion)(k) >= 0

and abs([RadarMotionMin](#RadarMotionMin)(k)) <= abs([MaximumSScompensatedMotion](#MaximumSScompensatedMotion)(k))

and abs([RadarMotionMax](#RadarMotionMax)(k)) >= abs([MinimumSScompensatedMotion](#MinimumSScompensatedMotion)(k)))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [RadarMotionMax](#RadarMotionMax) | √ | √ | NUMERIC\_32 |
| [RadarMotionMin](#RadarMotionMin) | √ | √ | NUMERIC\_32 |
| [RadarRawSpeed](#RadarRawSpeed) | √ | √ | NUMERIC\_32 |
| [RadarSpeedValid](#RadarSpeedValid) | √ | √ | BOOLEAN |

## F26-Detect Odometer Axle Lock

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATOcontrolTimeValid](#ATOcontrolTimeValid) | Internal | F11-Acquire Configuration Data |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [NonVitalRequest](#NonVitalRequest) | External | Interface with CC Non Vital |
| [OdometerState](#OdometerState) | Internal | F23-Manage the Odometer State |
| [RadarRawSpeed](#RadarRawSpeed) | Internal | F25-Calculate Radar Speed |
| [RadarSpeedValid](#RadarSpeedValid) | Internal | F25-Calculate Radar Speed |
| [ValidWheelKinematic](#ValidWheelKinematic) | Internal | F23-Manage the Odometer State |
| [WheelMinSpeed](#WheelMinSpeed) | Internal | F23-Manage the Odometer State |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [AxleLockedDetectionAvailable](#AxleLockedDetectionAvailable) | √ | √ | BOOLEAN |
| [AxlePossiblyLocked](#AxlePossiblyLocked) | √ | √ | BOOLEAN |
| [OdometerRef\_1](#OdometerRef_1) | √ | √ | ST\_ODOMETER\_REF |
| [OdometerRef\_2](#OdometerRef_2) | √ | √ | ST\_ODOMETER\_REF |
| [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold) | √ | √ | BOOLEAN |
| [ReferenceSpeedUnderThreshold\_1](#ReferenceSpeedUnderThreshold_1) | √ | √ | BOOLEAN |
| [ReferenceSpeedUnderThreshold\_2](#ReferenceSpeedUnderThreshold_2) | √ | √ | BOOLEAN |

### Processing

对于ATP，由于仅使用所在车头的里程计作为计算位移和速度的依据，因此必须防止该里程计所在的车轴由于制动而抱死，从而导致错误判断列车静止的危险。所以，需引入独立的两路参考速度，即当通过里程计测得“列车静止”，而参考速度显示“列车运动”时，即可判断出里程计所在车轴锁闭。两路参考速度信息OdometerRef\_1和OdometerRef\_2分别来自列车的测速系统，和远端ATP的里程计信息，其结构为ST\_ODOMETER\_REF：

As ATP calculate the movement and speed only based on the axle odometer assembled, it is necessary to avoid the situation that this axle is locked during the train braking, which may cause the danger that ATP judge the train status as stillness by mistake. Therefore, ATP need to adopt two independent sources of reference speed i.e. when the ATP detected the train is still through the odometer, while the referenced speed showed that the train is running, ATP can conclude that the wheel with odometer is blocked. The [OdometerRef\_1](#OdometerRef_1) and [OdometerRef\_2](#OdometerRef_2) are coming from the speed measuring system of the train and the odometer of the remote ATP respectively. The reference speed structure shows as below:

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_ODOMETER\_REF | |  |  |
|  | PossiblyDisabled | BOOLEAN | 该参考速度可能失效 |
|  | PossiblyEnabled | BOOLEAN | 该参考速度可能有效 |
|  | OutOfOrder | BOOLEAN | 该参考速度不可用 |
|  | Contradictory | BOOLEAN | 该参考速度与ATP测速不一致 |

#### Odometer speed availability

[iTC\_CC\_ATP-SwRS-0637]

OdometerSpeedAvailable，当前里程计测速是否可用于参考速度判断

[OdometerSpeedAvailable](#OdometerSpeedAvailable)(k):

return ([ValidWheelKinematic](#ValidWheelKinematic)(k)

and [OdometerState](#OdometerState)(k) is **INITIALIZED**)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-1153]

[End]

[iTC\_CC\_ATP-SwRS-0230]

OdometerSpeedUnderThreshold，本端里程计测速低于阈值。

ATP shall detect whether the measured wheel speed is under threshold.

def [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold)(k):

return ([WheelMinSpeed](#WheelMinSpeed)(k) < ATPsetting.OdoLockedAxleThresholdSpeed)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-1153], [iTC\_CC-SyAD-1155]

[End]

#### Obtaining reference speed

[iTC\_CC\_ATP-SwRS-0128]

ReferenceSpeedUnderThreshold\_1，来自CCNV的参考速度1是否小于指定阈值。

[ReferenceSpeedUnderThreshold\_1](#ReferenceSpeedUnderThreshold_1) defines whether the referenced speed 1 from CCNV is lower than a configurable threshold.

def [ReferenceSpeedUnderThreshold\_1](#ReferenceSpeedUnderThreshold_1)(k):

if [RadarSpeedValid](#RadarSpeedValid)(k):

return ([RadarRawSpeed](#RadarRawSpeed)(k) < [ATPsetting](#ATPsetting).OdoLockedAxleThresholdSpeed)

else:

return ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k)

and [NonVitalRequest](#NonVitalRequest).OdometerRef1SpeedUnderThreshold(k))

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1044], [iTC\_CC-SyAD-1153]

[End]

[iTC\_CC\_ATP-SwRS-0129]

ReferenceSpeedAvailable\_1，来自CCNV的参考速度1是否可用

[ReferenceSpeedAvailable\_1](#ReferenceSpeedAvailable_1) defines whether the referenced speed 1 from CCNV is valid or not.

def [ReferenceSpeedAvailable\_1](#ReferenceSpeedAvailable_1)(k):

return ([RadarSpeedValid](#RadarSpeedValid)(k)

or ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k)

and [NonVitalRequest](#NonVitalRequest).OdometerRef1Available(k)))

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1044], [iTC\_CC-SyAD-1153]

[End]

[iTC\_CC\_ATP-SwRS-0130]

ReferenceSpeedUnderThreshold\_2，来自CCNV的参考速度2是否小于指定阈值。

[ReferenceSpeedUnderThreshold\_2](#ReferenceSpeedUnderThreshold_2) defines whether the referenced speed 2 from CCNV is lower than a configurable threshold.

if ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k) == **True**)

[ReferenceSpeedUnderThreshold\_2](#ReferenceSpeedUnderThreshold_2)(k) = [NonVitalRequest](#NonVitalRequest).OdometerRef2SpeedUnderThreshold(k)

else:

[ReferenceSpeedUnderThreshold\_2](#ReferenceSpeedUnderThreshold_2) = **False**

#Category= Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1044], [iTC\_CC-SyAD-1153]

[End]

[iTC\_CC\_ATP-SwRS-0131]

ReferenceSpeedAvailable\_2，来自CCNV的参考速度2是否可用

[ReferenceSpeedAvailable\_2](#ReferenceSpeedAvailable_2) shows whether the referenced speed 2 from CCNV is effective or not.

if ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k) == **True**)

[ReferenceSpeedAvailable\_2](#ReferenceSpeedAvailable_2)(k) = [NonVitalRequest](#NonVitalRequest).OdometerRef2Available(k)

else:

[ReferenceSpeedAvailable\_2](#ReferenceSpeedAvailable_2) = **False**

#Category= Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1044], [iTC\_CC-SyAD-1153]

[End]

[iTC\_CC\_ATP-SwRS-0231]

[OdometerRef\_1](#OdometerRef_1).[PossiblyDisabled](#PossiblyDisabled)，当本端里程计可用且不为0速，而参考速度1可用但为0速时，则认为参考速度1可能错误

The independent source of odometry reference 1 said to disable if following conditions reached:

* local source of odometry is available ([ValidWheelKinematic](#ValidWheelKinematic)),
* and [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold) indicates that train speed is greater than reference speed threshold,
* and source of odometry reference 1 is available,
* and odometer reference 1 indicates that train speed is less than reference speed threshold.

[OdometerRef\_1](#OdometerRef_1). [PossiblyDisabled](#PossiblyDisabled)(k)

= ([ReferenceSpeedAvailable\_1](#ReferenceSpeedAvailable_1)(k)

and [ReferenceSpeedUnderThreshold\_1](#ReferenceSpeedUnderThreshold_1)(k)

and [OdometerSpeedAvailable](#OdometerSpeedAvailable)(k)

and not [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-1153]

[End]

[iTC\_CC\_ATP-SwRS-0232]

[OdometerRef\_2](#OdometerRef_2). [PossiblyDisabled](#PossiblyDisabled)，当本端里程计可用且不为0速，而参考速度2可用但为0速时，则认为参考速度2可能错误

The independent source of odometry reference 2 said to disable if following conditions reached:

* local source of odometry is available ([ValidWheelKinematic](#ValidWheelKinematic)),
* and [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold) indicates that train speed is greater than reference speed threshold,
* and source of odometry reference 2 is available,
* and odometer reference 2 indicates that train speed is less than reference speed threshold.

[OdometerRef\_2](#OdometerRef_2). [PossiblyDisabled](#PossiblyDisabled)(k)

= ([ReferenceSpeedAvailable\_2](#ReferenceSpeedAvailable_2)(k)

and [ReferenceSpeedUnderThreshold\_2](#ReferenceSpeedUnderThreshold_2)(k)

and [OdometerSpeedAvailable](#OdometerSpeedAvailable)(k)

and not [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-1153]

[End]

[iTC\_CC\_ATP-SwRS-0233]

[OdometerRef\_1](#OdometerRef_1).[PossiblyEnabled](#PossiblyEnabled)，当本端里程计和参考速度1均可用且测得列车在动时，认为参考速度1可能已恢复有效。

The independent source of odometry reference 1 said to enable if following conditions reached:

* local source of odometry is available,
* and [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold) indicates that train speed is greater than reference speed threshold,
* and source of odometry reference 1 is available,
* and odometer reference 1 indicates that train speed is greater than reference speed threshold.

[OdometerRef\_1](#OdometerRef_1). [PossiblyEnabled](#PossiblyEnabled)(k)

= ([ReferenceSpeedAvailable\_1](#ReferenceSpeedAvailable_1)(k)

and not [ReferenceSpeedUnderThreshold\_1](#ReferenceSpeedUnderThreshold_1)(k)

and [OdometerSpeedAvailable](#OdometerSpeedAvailable)(k)

and not [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-1153]

[End]

[iTC\_CC\_ATP-SwRS-0234]

[OdometerRef\_2](#OdometerRef_2).[PossiblyEnabled](#PossiblyEnabled)，当本端里程计和参考速度2均可用且测得列车在动时，认为参考速度2可能已恢复有效。

The independent source of odometry reference 2 said to enable if following conditions reached:

* local source of odometry is available,
* and [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold) indicates that train speed is greater than reference speed threshold,
* and source of odometry reference 2 is available,
* and odometer reference 2 indicates that train speed is greater than reference speed threshold.

[OdometerRef\_2](#OdometerRef_2). [PossiblyEnabled](#PossiblyEnabled)(k)

= ([ReferenceSpeedAvailable\_2](#ReferenceSpeedAvailable_2)(k)

and not [ReferenceSpeedUnderThreshold\_2](#ReferenceSpeedUnderThreshold_2)(k)

and [OdometerSpeedAvailable](#OdometerSpeedAvailable)(k)

and not [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-1153]

[End]

[iTC\_CC\_ATP-SwRS-0235]

[OdometerRef\_1](#OdometerRef_1).[OutOfOrder](#OutOfOrder)，当判断参考速度1可能不可用时，延迟一段时间，若仍不可用，则判断参考速度1失效。

The independent source of odometry reference 1 is said to be out of order if it is possibly disabled during more than [ATPsetting](#ATPsetting).OdoLockedAxleDisablingLatency.

if ([OdometerRef\_1](#OdometerRef_1).[PossiblyDisabled](#PossiblyDisabled)(k) == **True**

and [OdometerRef\_1](#OdometerRef_1).[PossiblyDisabled](#PossiblyDisabled)(k-1) == **True**

...

and [OdometerRef\_1](#OdometerRef_1).[PossiblyDisabled](#PossiblyDisabled)(k+1-[ATPsetting](#ATPsetting).OdoLockedAxleDisablingLatency) == **True**)

[OdometerRef\_1](#OdometerRef_1).OutOfOrder = **True**

当判断参考速度1可能可用时，延迟一段时间，若仍可用，则判断参考速度1有效

When the independent source of odometry reference 1 had out of order, it considered not out of order one if the source of odometry reference 1 is possibly enabled during more than [ATPsetting](#ATPsetting).OdoLockedAxleEnablingLatency:

if ([OdometerRef\_1](#OdometerRef_1).[PossiblyEnabled](#PossiblyEnabled)(k) == **True**

and [OdometerRef\_1](#OdometerRef_1).[PossiblyEnabled](#PossiblyEnabled)(k-1) == **True**

...

and [OdometerRef\_1](#OdometerRef_1).[PossiblyEnabled](#PossiblyEnabled)(k+1-[ATPsetting](#ATPsetting).OdoLockedAxleEnablingLatency) == **True**)

[OdometerRef\_1](#OdometerRef_1).[OutOfOrder](#OutOfOrder) = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC\_ATP\_SwHA-0237]

[End]

[iTC\_CC\_ATP-SwRS-0236]

[OdometerRef\_2](#OdometerRef_2).[OutOfOrder](#OutOfOrder)，当判断参考速度2可能不可用时，延迟一段时间，若仍不可用，则判断参考速度2失效。

The independent source of odometry reference 2 is said to be out of order if it is possibly disabled during more than [ATPsetting](#ATPsetting).OdoLockedAxleDisablingLatency.

if ([OdometerRef\_2](#OdometerRef_2).[PossiblyDisabled](#PossiblyDisabled)(k) == **True**

and [OdometerRef\_2](#OdometerRef_2).[PossiblyDisabled](#PossiblyDisabled)(k-1) == **True**

...

and [OdometerRef\_2](#OdometerRef_2).[PossiblyDisabled](#PossiblyDisabled)(k+1-[ATPsetting](#ATPsetting).OdoLockedAxleDisablingLatency) == **True**)

[OdometerRef\_2](#OdometerRef_2).[OutOfOrder](#OutOfOrder) = **True**

当判断参考速度2可能可用时，延迟一段时间，若仍可用，则判断参考速度2有效。

When the independent source of odometry reference 2 had out of order, It considered not out of order one if the source of odometry reference 2 is possibly enabled during more than [ATPsetting](#ATPsetting).OdoLockedAxleEnablingLatency:

if ([OdometerRef\_2](#OdometerRef_2).[PossiblyEnabled](#PossiblyEnabled)(k) == **True**

and [OdometerRef\_2](#OdometerRef_2).[PossiblyEnabled](#PossiblyEnabled)(k-1) == **True**

...

and [OdometerRef\_2](#OdometerRef_2).[PossiblyEnabled](#PossiblyEnabled)(k+1-[ATPsetting](#ATPsetting).OdoLockedAxleEnablingLatency) == **True**)

[OdometerRef\_2](#OdometerRef_2).OutOfOrder = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC\_ATP\_SwHA-0237]

[End]

[iTC\_CC\_ATP-SwRS-0237]

[OdometerRef\_1](#OdometerRef_1).[Contradictory](#Contradictory)，若参考速度1有效且判断车动，而本端里程计判断车静止，则认为参考速度1判断出里程计可能故障。

The source of odometry reference 1 said to be contradictory with local source of odometry if:

* local source of odometry is available ([ValidWheelKinematic](#ValidWheelKinematic))
* and [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold) indicates that wheel speed is less than reference speed threshold,
* and source of odometry reference 1 is available and not out of order,
* and odometer reference 1 indicates that train speed is greater than reference speed threshold.

[OdometerRef\_1](#OdometerRef_1). [Contradictory](#Contradictory)(k)

= (not [OdometerRef\_1](#OdometerRef_1).OutOfOrder(k)

and [ReferenceSpeedAvailable\_1](#ReferenceSpeedAvailable_1)(k)

and not [ReferenceSpeedUnderThreshold\_1](#ReferenceSpeedUnderThreshold_1)(k)

and [ValidWheelKinematic](#ValidWheelKinematic)(k)

and [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-1156], [iTC\_CC\_ATP\_SwHA-0091]

[End]

[iTC\_CC\_ATP-SwRS-0238]

[OdometerRef\_2](#OdometerRef_2). [Contradictory](#Contradictory)，若参考速度2有效且判断车动，而本端里程计判断车静止，则认为参考速度2判断出里程计可能故障。

The source of odometry reference 2 said to be contradictory with local source of odometry if:

* local source of odometry is available ([ValidWheelKinematic](#ValidWheelKinematic))
* and [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold) indicates that wheel speed is less than reference speed threshold,
* and source of odometry reference 2 is available and not out of order,
* and odometer reference 2 indicates that train speed is greater than reference speed threshold.

[OdometerRef\_2](#OdometerRef_2).[Contradictory](#Contradictory) (k)

= (not [OdometerRef\_2](#OdometerRef_2).OutOfOrder(k)

and [ReferenceSpeedAvailable\_2](#ReferenceSpeedAvailable_2)(k)

and not [ReferenceSpeedUnderThreshold\_2](#ReferenceSpeedUnderThreshold_2)(k)

and [ValidWheelKinematic](#ValidWheelKinematic)(k)

and [OdometerSpeedUnderThreshold](#OdometerSpeedUnderThreshold)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-1156], [iTC\_CC\_ATP\_SwHA-0091]

[End]

#### Axle locked detection

[iTC\_CC\_ATP-SwRS-0239]

AxlePossiblyLocked，在两路参考速度都正常（没有失效out of order）的情况下，当两路参考速度均判断本端里程计可能故障的情况下，认为当前可能轴锁。或者，当有一路参考速度认为轴锁，而另一路参考速度失效或不可用，也认为当前可能轴锁。

Odometer axle shall consider possibly locked if:

* Both independent sources of odometry indicates a contradiction with local odometer,
* Or one source of odometry is contradictory and the other one is out of order (or not available).

[AxlePossiblyLocked](#AxlePossiblyLocked)(k)

= (([OdometerRef\_1](#OdometerRef_1).[Contradictory](#Contradictory)(k) and [OdometerRef\_2](#OdometerRef_2).[Contradictory](#Contradictory)(k))

or ([OdometerRef\_1](#OdometerRef_1).[Contradictory](#Contradictory)(k)

and ([OdometerRef\_2](#OdometerRef_2).[OutOfOrder](#OutOfOrder)(k) or not [ReferenceSpeedAvailable\_2](#ReferenceSpeedAvailable_2)(k)))

or ([OdometerRef\_2](#OdometerRef_2).[Contradictory](#Contradictory)(k)

and ([OdometerRef\_1](#OdometerRef_1).[OutOfOrder](#OutOfOrder)(k) or not [ReferenceSpeedAvailable\_1](#ReferenceSpeedAvailable_1)(k))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-1156], [iTC\_CC\_ATP\_SwHA-0091]

[End]

[iTC\_CC\_ATP-SwRS-0240]

UnrecoverableAxleLocked，当连续若干个周期判断可能轴锁，或者已经判断为轴锁，则永久轴锁.

If [AxlePossiblyLocked](#AxlePossiblyLocked) situation lasts more than [ATPsetting](#ATPsetting).OdoLockedAxleTimeout, the odometer axle shall be considered locked. Once [UnrecoverableAxleLocked](#UnrecoverableAxleLocked) set to **True**, it will stay at state **True** unless ATP re-initialized.

[UnrecoverableAxleLocked](#UnrecoverableAxleLocked)(k)

= [UnrecoverableAxleLocked](#UnrecoverableAxleLocked)(k-1)

or ([AxlePossiblyLocked](#AxlePossiblyLocked)(k)

and [AxlePossiblyLocked](#AxlePossiblyLocked)(k-1)

and ...

and [AxlePossiblyLocked](#AxlePossiblyLocked)(k+1-[ATPsetting](#ATPsetting).OdoLockedAxleTimeout)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-0137], [iTC\_CC-SyAD-0364], [iTC\_CC\_ATP\_SwHA-0091]

[End]

[iTC\_CC\_ATP-SwRS-0241]

AxleLockedDetectionAvailable，只要有一路参考速度可以工作，就认为轴锁侦测可用。

If only one or no source of odometry is available, then ATP shall invalidate kinematic while this situation lasting.

[AxleLockedDetectionAvailable](#AxleLockedDetectionAvailable)

= ((not [OdometerRef\_1](#OdometerRef_1).[OutOfOrder](#OutOfOrder) and [ReferenceSpeedAvailable\_1](#ReferenceSpeedAvailable_1)(k))

or (not [OdometerRef\_2](#OdometerRef_2).[OutOfOrder](#OutOfOrder) and [ReferenceSpeedAvailable\_2](#ReferenceSpeedAvailable_2)(k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-1157], [iTC\_CC\_ATP\_SwHA-0091], [iTC\_CC\_ATP\_SwHA-0237], [iTC\_CC\_ATP\_SwHA-0238]

[End]

[iTC\_CC\_ATP-SwRS-0242]

WheelTrainKinematicCorrelation，车轮和列车的速度一致性

Wheel and train kinematic shall consider correctly correlated if and only if:

* odometer axle is not detected locked,
* and odometer axle detection is available

[WheelTrainKinematicCorrelation](#WheelTrainKinematicCorrelation)(k)

= [AxleLockedDetectionAvailable](#AxleLockedDetectionAvailable)(k) and not [UnrecoverableAxleLocked](#UnrecoverableAxleLocked)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-0364], [iTC\_CC-SyAD-0137], [iTC\_CC-SyAD-1157], [iTC\_CC\_ATP\_SwHA-0238]

[End]

### Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | | Safety Critical | | Observable | | Logical Type |
| [OdometerSpeedAvailable](#OdometerSpeedAvailable) | √ | | √ | | BOOLEAN | |
| [UnrecoverableAxleLocked](#UnrecoverableAxleLocked) | √ | | √ | | BOOLEAN | |
| [WheelTrainKinematicCorrelation](#WheelTrainKinematicCorrelation) | √ | | √ | | BOOLEAN | |

## F27-Compute Train Kinematics

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [BeaconMessageReceive](#BeaconMessageReceive) | Internal | F21-Manage Interface with |
| [CogPositionAfterToploc](#CogPositionAfterToploc) | Internal | F22-Monitor the Odometer |
| [CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc) | Internal | F22-Monitor the Odometer |
| [LockedBeaconMsgByte](#LockedBeaconMsgByte) | Internal | F21-Manage Interface with |
| [MaxCogCalibration](#MaxCogCalibration) | Internal | F28-Calibrate Wheel Movement |
| [[MaximumSScompensatedMotion](#MaximumSScompensatedMotion)](#OverestimatedMotionMax) | Internal | F24-Compensate Sliding Slipping Effect |
| [MinCogCalibration](#MinCogCalibration) | Internal | F28-Calibrate Wheel Movement |
| [MinimumSScompensatedMotion](#MinimumSScompensatedMotion) | Internal | F24-Compensate Sliding Slipping Effect |
| [MotionOverEstimationState](#MotionOverEstimationState) | Internal | F24-Compensate Sliding Slipping Effect |
| [OdometerSpeedAvailable](#OdometerSpeedAvailable) | Internal | F26-Detect Odometer Axle Lock |
| [RadarMotionMax](#RadarMotionMax) | Internal | F25-Calculate Radar Speed |
| [RadarMotionMin](#RadarMotionMin) | Internal | F25-Calculate Radar Speed |
| [RadarSpeedValid](#RadarSpeedValid) | Internal | F25-Calculate Radar Speed |
| [TeethCounter](#TeethCounter) | Internal | F22-Monitor the Odometer |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [ValidSlipSlideModelling](#ValidSlipSlideModelling) | Internal | F24-Compensate Sliding Slipping Effect |
| [ValidWheelKinematic](#ValidWheelKinematic) | Internal | F23-Manage the Odometer State |
| [WheelFilteredStopped](#WheelFilteredStopped) | Internal | F22-Monitor the Odometer |
| [WheelStopped](#WheelStopped) | Internal | F22-Monitor the Odometer |
| [WheelTrainKinematicCorrelation](#WheelTrainKinematicCorrelation) | Internal | F26-Detect Odometer Axle Lock |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [TrainMinSpeed](#TrainMinSpeed) | √ | √ | NUMERIC\_32 |
| [TrainStopped](#TrainStopped) | √ | √ | BOOLEAN |

### Processing

来自里程计测得的车轮运动学参数，经过打滑补偿和轴锁判断处理后，即可得到安全的列车运动学参数，供ATP的其他功能使用。

After the overestimation of the sliding effect and consideration of wheel braking decocted by odometers, ATP can calculate the train kinematic parameters which will used by other functions.

对于配置有多普勒雷达的项目，可用雷达测速值修正里程计受所在车轮打滑、空转的影响，提高运营效率。里程计和雷达的融合方案如下：

* 当雷达不可用，仅使用里程计的位移值；
* 若判断里程计打滑控制模型失效（SKIDDING状态），则使用雷达的位移值；
* 其他情况，使用雷达和里程计最大最小位移的交集。

[iTC\_CC\_ATP-SwRS-0243]

ValidTrainKinematic，列车位移速度计算有效的条件

Train kinematic information shall declare invalid if at least one of following condition is **True**:

* odometer kinematic is not valid,
* or neither radar speed nor over-estimation modeling of train movement is valid,
* or train movement and wheel movement has been detected de-correlated.

def [ValidTrainKinematic](#ValidTrainKinematic)(k):

return (([ValidSlipSlideModelling](#ValidSlipSlideModelling)(k)

or [RadarSpeedValid](#RadarSpeedValid)(k))

and [WheelTrainKinematicCorrelation](#WheelTrainKinematicCorrelation)(k)

and [ValidWheelKinematic](#ValidWheelKinematic)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-0186], [iTC\_CC-SyAD-0192], [iTC\_CC-SyAD-0364], [iTC\_CC-SyAD-0960], [iTC\_CC-SyAD-0137], [iTC\_CC-SyAD-1135], [iTC\_CC-SyAD-1159], [iTC\_CC\_ATP\_SwHA-0091], [iTC\_CC\_ATP\_SwHA-0092], [iTC\_CC\_ATP\_SwHA-0238]

[End]

[iTC\_CC\_ATP-SwRS-0212]

MaximumTrainMotion，根据来自里程计或雷达的信息，计算列车的周期最大位移。该值为矢量，向**END\_1**方向为正，**END\_2**方向为负。

def [MaximumTrainMotion](#MaximumTrainMotion)(k):

if (not [RadarSpeedValid](#RadarSpeedValid)(k)):

return [MaximumSScompensatedMotion](#MaximumSScompensatedMotion)(k)

elif (not [ValidSlipSlideModelling](#ValidSlipSlideModelling)(k)):

return [RadarMotionMax](#RadarMotionMax)(k)

else:

return (min(abs([MaximumSScompensatedMotion](#MaximumSScompensatedMotion)(k)), abs([RadarSpeedMax](#RadarSpeedMax)(k)))

\* sign([MaximumSScompensatedMotion](#MaximumSScompensatedMotion)(k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=

[End]

[iTC\_CC\_ATP-SwRS-0736]

MinimumTrainMotion，根据来自里程计或雷达的信息，计算列车的周期最小位移。该值为矢量，向**END\_1**方向为正，**END\_2**方向为负。

def [MinimumTrainMotion](#MinimumTrainMotion)(k):

if (not [RadarSpeedValid](#RadarSpeedValid)(k)):

return [MinimumSScompensatedMotion](#MinimumSScompensatedMotion)(k)

elif (not [ValidSlipSlideModelling](#ValidSlipSlideModelling)(k)):

return [RadarMotionMin](#RadarMotionMin)(k)

else:

return (max(abs([[MinimumSScompensatedMotion](#MinimumSScompensatedMotion)](#OverestimatedMotionMin)(k)), abs([RadarSpeedMin](#RadarSpeedMin)(k)))

\* sign([[MinimumSScompensatedMotion](#MinimumSScompensatedMotion)](#OverestimatedMotionMin)(k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=

[End]

[iTC\_CC\_ATP-SwRS-0244]

TrainStopped, train shall consider strictly stopped if and only if:

* wheel is detected strictly stopped,
* and train kinematic elaboration is valid,
* and wheel is not detected sliding.

[TrainStopped](#TrainStopped)(k)

= (([WheelStopped](#WheelStopped)(k) == **True**)

and (([MotionOverEstimationState](#MotionOverEstimationState) == **COASTING**)

or ([MotionOverEstimationState](#MotionOverEstimationState) == **BRAKING**))

and ([ValidTrainKinematic](#ValidTrainKinematic)(k) == **True**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0150], [iTC\_CC-SyAD-0364], [iTC\_CC\_ATP\_SwHA-0093]

[End]

[iTC\_CC\_ATP-SwRS-0245]

TrainFilteredStopped，列车准静止判断。

Train shall consider stopped with the tolerance of one cog detection if:

* wheel is detected at filtered stop,
* and train kinematic elaboration is valid,
* and wheel is not detected sliding.

[TrainFilteredStopped](#TrainFilteredStopped)(k)

= (([WheelFilteredStopped](#WheelFilteredStopped)(k) == **True**)

and (([MotionOverEstimationState](#MotionOverEstimationState) == **COASTING**)

or ([MotionOverEstimationState](#MotionOverEstimationState) == **BRAKING**))

and ([ValidTrainKinematic](#ValidTrainKinematic)(k) == **True**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0151], [iTC\_CC-SyAD-0214], [iTC\_CC-SyAD-0364], [iTC\_CC-SyAD-0152], [iTC\_CC\_ATP\_SwHA-0094]

[End]

[iTC\_CC\_ATP-SwRS-0638]

TrainHasMoved，表明自上电以后，列车是否移动过。

def [TrainHasMoved](#TrainHasMoved)(k):

if (Initialization):

return **False**

elif (not [TrainHasMoved](#TrainHasMoved)(k-1)

and not [TrainFilteredStopped](#TrainFilteredStopped)(k)

and ([TeethCounter](#TeethCounter)(k) != [TeethCounter](#TeethCounter)(k-1))):

return **True**

else:

return [TrainHasMoved](#TrainHasMoved)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1204], [iTC\_CC\_ATP\_SwHA-0239]

[End]

[iTC\_CC\_ATP-SwRS-0639]

TrainMinSpeed，计算列车最小速度。

def TrainMinSpeed(k):

if ([OdometerSpeedAvailable](#OdometerSpeedAvailable)(k)):

return [max](#max)(0, ([round.floor](#roundfloor)([abs](#abs)([MinimumTrainMotion](#MinimumTrainMotion)(k) / ATP\_CYCLE\_TIME))

+ (([ATPsetting](#ATPsetting).BrakingMinAcc - [ATPsetting](#ATPsetting).MaxGradientAcc)

\* ATP\_CYCLE\_TIME / 2)))

else:

return 0

其中：

* [ATPsetting](#ATPsetting).BrakingMinAcc，为列车制动最小减速度，为负值（即绝对值最大）；
* [ATPsetting](#ATPsetting).MaxGradientAcc，为线路中最大坡度加速度，为正值。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0146]

[End]

[iTC\_CC\_ATP-SwRS-0246]

TrainMaxSpeed，考虑打滑过估补偿的列车最大速度，该速度为非负值。

According to the matching of odometer cog counter and code, maximum train speed shall computed using the followings expressions:

def [TrainMaxSpeed](#TrainMaxSpeed)(k):

if ([OdometerSpeedAvailable](#OdometerSpeedAvailable)(k)):

return ([round.ceil](#roundceil)([abs](#abs)([MaximumTrainMotion](#MaximumTrainMotion)(k) / **ATP\_CYCLE\_TIM**E))

+ (([ATPsetting](#ATPsetting).TractionMaxAcc[[TrainMinSpeed](#TrainMinSpeed)(k)] + [ATPsetting](#ATPsetting).MaxGradientAcc)

\* **ATP\_CYCLE\_TIME** / 2))

else:

return [round.ceil](#roundceil)([abs](#abs)([MaximumTrainMotion](#MaximumTrainMotion)(k)) / **ATP\_CYCLE\_TIME**)

其中：

* [ATPsetting](#ATPsetting).TractionMaxAcc，表示列车在不同车速下的最大牵引力加速度

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0146], [iTC\_CC-SyAD-0217], [iTC\_CC\_ATP\_SwHA-0095]

[End]

**NOTES:**

如果[TrainMinSpeed](#TrainMinSpeed)已经大于配置数据[ATPsetting](#ATPsetting).TractionMaxAcc表中的最大速度值，则应当取[ATPsetting](#ATPsetting).TractionMaxAcc中设置的最大速度所对应的牵引加速度，作为当前计算使用的最大牵引加速度。根据车辆牵引特性，高速情况牵引能力随着速度的增加而减小，因此，取比当前车速低的速度所对应的牵引加速度是导向安全的。

[iTC\_CC\_ATP-SwRS-0640]

NewBeaconObtained，表明收到了可用的RB

def [NewBeaconObtained](#NewBeaconObtained)(k):

if ([BeaconMessageReceive](#BeaconMessageReceive)(k)

and [ValidTrainKinematic](#ValidTrainKinematic)(k)

and not [TrainFilteredStopped](#TrainFilteredStopped)(k)

and [OdometerState](#OdometerState)(k) is **INITIALIZED**):

[NewBeaconObtained](#NewBeaconObtained) = **True**

else:

[NewBeaconObtained](#NewBeaconObtained) = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0176], [iTC\_CC-SyAD-0177], [iTC\_CC-SyAD-0196]

[End]

[iTC\_CC\_ATP-SwRS-0641]

BeaconBeforeLastObtained，记录读到的次新的信标

def [BeaconBeforeLastObtained](#BeaconBeforeLastObtained)(k):

if (Initialization):

[BeaconBeforeLastObtained](#BeaconBeforeLastObtained) = **None**

elif ([NewBeaconObtained](#NewBeaconObtained)(k)):

[BeaconBeforeLastObtained](#BeaconBeforeLastObtained) = [BeaconLastObtained](#BeaconLastObtained)(k-1)

else:

[BeaconBeforeLastObtained](#BeaconBeforeLastObtained) = [BeaconBeforeLastObtained](#BeaconBeforeLastObtained)(k-1)

return [BeaconBeforeLastObtained](#BeaconBeforeLastObtained)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0176], [iTC\_CC-SyAD-0177], [iTC\_CC-SyAD-0196]

[End]

[iTC\_CC\_ATP-SwRS-0642]

BeaconLastObtained，记录读到的最新的信标

def [BeaconLastObtained](#BeaconLastObtained)(k):

if (Initialization):

[BeaconLastObtained](#BeaconLastObtained) = **None**

elif ([NewBeaconObtained](#NewBeaconObtained)(k)):

[BeaconLastObtained](#BeaconLastObtained) = [TrackMap](#TrackMap).[Beacons](#Beacons)[[LockedBeaconMsgByte](#LockedBeaconMsgByte).Id]

else:

[BeaconLastObtained](#BeaconLastObtained) = [BeaconLastObtained](#BeaconLastObtained)(k-1)

return [BeaconLastObtained](#BeaconLastObtained)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0176], [iTC\_CC-SyAD-0177], [iTC\_CC-SyAD-0196]

[End]

[iTC\_CC\_ATP-SwRS-0247]

在本周期的里程计和测速信息有效的情况下，需要根据当前齿数和锁存的读到信标时的Top-loc信息，计算DistLastBeaconMax和DistLastBeaconMin，表示当前经过信标后已运行的最大最小距离。

If a valid beacon with top-loc received between cycle k-1 and k, then minimum and maximum distance ran since top-loc shall evaluate as the difference between current teeth counter and recorded cog position just before or after top-loc. If there is no beacon received, ATP updates distances from last beacon using the train movements.

if ([NewBeaconObtained](#NewBeaconObtained)(k)):

[DistLastBeaconMin](#DistLastBeaconMin)(k)= [MinCogCalibration](#MinCogCalibration)(k) \* ([TeethCounter](#TeethCounter)(k)- [CogPositionAfterTopLoc](#CogPositionAfterTopLoc)(k))

[DistLastBeaconMax](#DistLastBeaconMax)(k)= [MaxCogCalibration](#MaxCogCalibration)(k) \* ([TeethCounter](#TeethCounter)(k)- [CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc)(k))

else:

[DistLastBeaconMin](#DistLastBeaconMin) = [DistLastBeaconMin](#DistLastBeaconMin)(k-1) + [MinimumTrainMotion](#MinimumTrainMotion)(k)

[DistLastBeaconMax](#DistLastBeaconMax) = [DistLastBeaconMax](#DistLastBeaconMax)(k-1) + [MaximumTrainMotion](#MaximumTrainMotion)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0182], [iTC\_CC\_ATP\_SwHA-0097], [iTC\_CC-SyAD-0183]

[End]

**NOTES:**

根据里程计状态和列车位移方向，对列车运行方向的判断如下：

* 如果里程计还未初始化，则不能判断出正确的里程计转动方向，即认为列车既向**END\_1**也向**END\_2**方向运行；
* 否则，当里程计初始化后：
* 如果列车最小位移为0，认为列车既不向**END\_1**也不向**END\_2**方向运行；
* 如果列车最小位移大于0，则认为列车向**END\_1**方向运行；
* 如果列车最小位移小于0，则认为列车向**END\_2**方向运行。

[iTC\_CC\_ATP-SwRS-0635]

End2RunningForward，根据车轮旋转方向，判断列车是否向**END\_2**方向运行

def [End2RunningForward](#End2RunningForward)(k):

if ([OdometerState](#OdometerState)(k) is **INITIALIZED**):

return ([MaximumTrainMotion](#MaximumTrainMotion)(k) < 0)

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_SyAD-0135], [iTC\_CC\_SyAD-0204], [iTC\_CC\_ATP\_SwHA-0277]

[End]

[iTC\_CC\_ATP-SwRS-0758]

End1RunningForward，根据车轮旋转方向，判断列车是否向**END\_1**方向运行

def [End1RunningForward](#End1RunningForward)(k):

if ([OdometerState](#OdometerState)(k) is **INITIALIZED**):

return ([MaximumTrainMotion](#MaximumTrainMotion)(k) > 0)

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_SyAD-0135], [iTC\_CC\_SyAD-0204], [iTC\_CC\_ATP\_SwHA-0277]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [BeaconBeforeLastObtained](#BeaconBeforeLastObtained) | √ | √ | NUMERIC\_32 |
| [BeaconLastObtained](#BeaconLastObtained) | √ | √ | NUMERIC\_32 |
| [DistLastBeaconMax](#DistLastBeaconMax) | √ | √ | NUMERIC\_32 |
| [DistLastBeaconMin](#DistLastBeaconMin) | √ | √ | NUMERIC\_32 |
| [End1RunningForward](#End1RunningForward) | √ | √ | NUMERIC\_32 |
| [End2RunningForward](#End2RunningForward) | √ | √ | NUMERIC\_32 |
| [MaximumTrainMotion](#MaximumTrainMotion) | √ | √ | NUMERIC\_32 |
| [MinimumTrainMotion](#MinimumTrainMotion) | √ | √ | NUMERIC\_32 |
| [NewBeaconObtained](#NewBeaconObtained) | √ | √ | BOOLEAN |
| [TrainFilteredStopped](#TrainFilteredStopped) | √ | √ | BOOLEAN |
| [TrainHasMoved](#TrainHasMoved) | √ | √ | BOOLEAN |
| [TrainMaxSpeed](#TrainMaxSpeed) | √ | √ | NUMERIC\_32 |
| [ValidTrainKinematic](#ValidTrainKinematic) | √ | √ | BOOLEAN |

## F28-Calibrate Wheel Movement

编码里程计安装在车轴上，其测试码盘外圈均匀分布着100个齿，ATP根据编码里程计传感器检测到齿数的脉冲，估算出列车的位移。在使用前，需对每个齿所代表的长度，即齿距进行校准。校准通过线路地图上相邻的一对固定间距的MTIB，及其后紧跟的验证RB完成。

The coding odometer, assembled on axle, has 100 cogs evenly distributed in the outer ring. According to the cog impulse detected by sensor from coded odometer, ATP will estimate the train movement. Therefore, ATP needs to calibrate the cog length by the adjacent MTIB and a verification RB on the track.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [BeaconMessage](#BeaconMessage) | Internal | F21-Manage Interface with |
| [CogPositionAfterTopLoc](#CogPositionAfterTopLoc) | Internal | F22-Monitor the Odometer |
| [CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc) | Internal | F22-Monitor the Odometer |
| [[End1RunningForward](#End1RunningForward)](#End2RunningForward) | Internal | F27-Compute Train Kinematics |
| [NewBeaconObtained](#NewBeaconObtained) | Internal | F27-Compute Train Kinematics |
| [SlipSlideDetected](#SlipSlideDetected) | Internal | F24-Compensate Sliding Slipping Effect |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [ValidTrainKinematic](#ValidTrainKinematic) | Internal | F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [CalibrationState](#CalibrationState) | √ | √ | ENUM\_CALIBRATION\_STATE |

### Processing

里程计外圈均匀分布着100个齿，每个齿的齿距就代表着车轮周长的1/100，因此齿距可以通过列车车轮的轮径计算出来。车辆属性中理论的最大最小轮径值就对应着理论的最大最小齿距值。在刚上电时，ATP根据理论最大最小齿距计算列车位移。当列车通过线路上一对固定距离（该距离描述在线路地图中）的信标来对实际齿距进行计算，并将结果通过第三个信标进行检验。如果检验成功，则ATP使用该齿距值作为运动学计算的依据。

There are 100 cogs evenly distributed in the outer ring of odometer, and each cog’s length represented the 1/100 of wheel perimeter, so we can calculate the cog’s length by the wheel diameter. Theoretically, the biggest and the smallest wheel diameter is corresponded to the cog length. When ATP is powered up, it will calculate the train displacement according to the theoretical cog length. When the train passed the fixed distance (this distance described in the track map), ATP will calibrate the real cog length according to the beacon of this fixed distance, and verify this calibration through the third beacon. If the calculation is correct, ATP will use this cog length for the kinematic calculation.

[iTC\_CC\_ATP-SwRS-0192]

如Figure 5‑10所示，里程计齿距校准CalibrationState分为以下四个状态:

* **CALI\_WAITING**, 未校准或者未成功校准，等待经过MTIB1后进行校准的状态;
* **CALI\_MEASURING**, 经过MTIB1,还未到MTIB2, 正在校准的状态；
* **CALI\_VALIDATING**, 经过MTIB2，还未到下个验证RB的状态；
* **CALI\_COMPLETED**, 经过验证信标RB，并验证成功，本次校准完成。

As shown in Figure 5‑10, the odometer calibration state divides into the following four states:

* **CALI\_WAITING**, the state of not calibrated or calibration has failed, waiting for read the MTIB1 beacon.
* **CALI\_MEASURING**, after read MTIB1 and waiting for read MTIB2.
* **CALI\_VALIDATING**, after read MTIB2 and waiting for the verifying RB.
* **CALI\_COMPLETED**, calibration has been verified by the RB successfully.

齿距校准过程中需更新下列信息：

* MaxCogCalibration, 最大齿距
* MinCogCalibration, 最小齿距
* [CalibrationState](#CalibrationState), 齿距校准状态

The following variables shall update during calibration:

* [MaxCogCalibration](#MaxCogCalibration), the overestimated calibration for each cog;
* [MinCogCalibration](#MinCogCalibration), the underestimated calibration for each cog;
* [CalibrationState](#CalibrationState), the state of calibration.

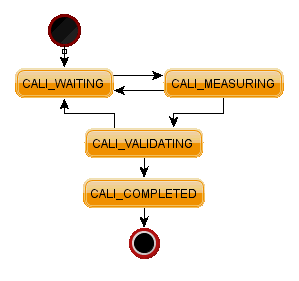


Figure 5‑10 Processing of calibration

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0132], [iTC\_CC\_ATP\_SwHA-0067]

[End]

#### State “**CALI\_WAITING**”

[iTC\_CC\_ATP-SwRS-0193]

在**CALI\_WATING**状态下使用默认齿距值

From power-up and while calibration process is not successfully performed, ATP shall use default calibration to compute train motion and shall consider itself in the state of waiting for the first beacon belonging to a couple of calibration.

if (Initialization)

[CalibrationState](#CalibrationState) = **CALI\_WAITING**

[MinCogCalibration](#MinCogCalibration) = [ATPsetting](#ATPsetting).OdoCaliDefaultCogLengthMin

[MaxCogCalibration](#MaxCogCalibration) = [ATPsetting](#ATPsetting).OdoCaliDefaultCogLengthMax

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0132], [iTC\_CC-SyAD-0139], [iTC\_CC\_ATP\_SwHA-0195]

[End]

[iTC\_CC\_ATP-SwRS-0463]

当读到线路地图中的MTIB1时，齿距校准状态从**CALI\_WAITING**转入**CALI\_MEASURING**。

If ATP is in the state of **CALI\_WAITING**, can transform to the measuring state if following conditions fulfilled:

* a valid beacon has been received and this beacon belongs to a couple of calibration,
* and train kinematic was valid,
* and no excessive slip/slide effect was detected,
* and [WheelMinimumMovement](#WheelMinimumMovement) is not null,

Then, ATP shall memorize:

* position of the wheel before and after top location signal of received beacon
* the ID of received beacon,
* the sign of the movement when crossing beacon,
* and shall consider itself as **CALI\_MEASURING**.

if (([CalibrationState](#CalibrationState)(k-1) = **CALI\_WAITING**)

and ([NewBeaconObtained](#NewBeaconObtained)(k) == **True**)

and ([TrackMap](#TrackMap).[BeaconBelongsToCalibrationCouple](#BeaconBelongsToCalibrationCouple)([BeaconMessage](#BeaconMessage).Id(k)) == **True**)

and ([SlipSlideDetected](#SlipSlideDetected)(k) == **False**))

CalibrationMeasurementStartPositionMin = [CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc)(k)

CalibrationMeasurementStartPositionMax = [CogPositionAfterTopLoc](#CogPositionAfterTopLoc)(k)

CalibrationMeasurementStartBeacon = [BeaconMessage](#BeaconMessage).Id(k)

CalibrationEnd1RunningForward = [End1RunningForward](#End1RunningForward)(k)

[CalibrationState](#CalibrationState) = **CALI\_MEASURING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0140], [iTC\_CC-SyAD-0141], [iTC\_CC-SyAD-0145], [iTC\_CC-SyAD-0183]

[End]

#### State “**CALI\_MEASURING**”

[iTC\_CC\_ATP-SwRS-0194]

当读到线路地图中与之前MTIB1匹配的MTIB2时，根据经过的齿数值，在[ATPsetting](#ATPsetting)中查表得到最大最小齿距，并比较测得的齿距结果：

* 如果测得的最大最小齿距在理论值范围内，则齿距校准状态从**CALI\_MEASURING**到**CALI\_VALIDATING**；
* 如果在理论范围外，则状态从**CALI\_MEASURING**转入**CALI\_WAITING**。

If ATP is in the state of **CALI\_MEASURING and** following conditions fulfilled:

* a valid beacon has been received and the beacon and first memorized calibration beacon is one of possible dedicated couple of calibration,
* and train kinematic was valid,
* and no excessive slip/slide effect was detected,
* and sign of train motion is still identical to thus detected on first beacon signaling,

Then,

* if resulting calibration range is included in default calibration range, then ATP shall:
* memorize:
* position of the wheel before and after top location signal of received beacon,
* the id of received beacon,
* and shall consider itself as **CALI\_VALIDATING**.
* else: ATP shall consider that calibration process has failed and back to **CALI\_WAITING**.

if (([CalibrationState](#CalibrationState)(k-1) == **CALI\_MEASURING)**

and ([NewBeaconObtained](#NewBeaconObtained)(k) == **True**)

and ([TrackMap](#TrackMap).[AreNeighbouredBeacons](#AreNeighbouredBeacons)([BeaconMessage](#BeaconMessage).Id(k),

CalibrationMeasurementStartBeacon(k)) == **True**)

and ([SlipSlideDetected](#SlipSlideDetected)(k) == **False**)

and ([CalibrationEnd1RunningForward](#CalibrationEnd1RunningForward)(k-1) == [End1RunningForward](#End1RunningForward)(k)))

if (([ATPsetting](#ATPsetting).OdoCaliDefaultCogLengthMax >= [MaxCogCalibrationMeasured](#MaxCogCalibrationMeasured))

and ([ATPsetting](#ATPsetting).OdoCaliDefaultCogLengthMin <= [MinCogCalibrationMeasured](#MinCogCalibrationMeasured)))

CalibrationValidationStartPositionMin = [CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc)(k)

CalibrationValidationStartPositionMax = [CogPositionAfterTopLoc](#CogPositionAfterTopLoc)(k)

CalibrationValidationStartBeacon = [BeaconMessage](#BeaconMessage).Id(k))

CalibrationResultMin = [MinCogCalibrationMeasured](#MinCogCalibrationMeasured)

CalibrationResultMax = [MaxCogCalibrationMeasured](#MaxCogCalibrationMeasured)

[CalibrationState](#CalibrationState) = **CALI\_VALIDATING**

else:

[CalibrationState](#CalibrationState) = **CALI\_WAITING**

其中最大最小测得齿距是根据校准过程中测得齿数，在离线工具计算的齿数-齿距对照表中查到的：

Among them, the measured maximum and minimum calibration fetches from the offline-generated counter-calibration table, based on the calculated cog counter.

MinCogCalibrationMeasured

= CaliMinRatio \* [ATPsetting](#ATPsetting).MeterCaliMaxMinCalibration[1]

[|[CalibrationMeasurementStartPositionMin](#CalibrationMeasurementStartPositionMin)(k)-[CogPositionAfterTopLoc](#CogPositionAfterTopLoc)(k)|

- [ATPsetting](#ATPsetting).OdoCaliCogCounterMin]

MaxCogCalibrationMeasured

= CaliMaxRatio \* [ATPsetting](#ATPsetting).MeterCaliMaxMinCalibration[0]

[|[CalibrationMeasurementStartPositionMax](#CalibrationMeasurementStartPositionMax)(k)-[CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc)(k)|

- [ATPsetting](#ATPsetting).OdoCaliCogCounterMin]

其中CaliMinRatio和CaliMaxRatio是离线工具计算的该校准信标所对应的MTIB结构的相关属性，表示该对MTIB的间距与标准间距（21米）的比率，该结构定义见[REF11]。

The CaliMinRation and CaliMaxRation are elements of structure MTIB generated by offline tool for each couple of calibration beacons. Refer to [REF11] for the definition of MTIB.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0132], [iTC\_CC-SyAD-0143], [iTC\_CC-SyAD-0144], [iTC\_CC-SyAD-0145], [iTC\_CC-SyAD-0183], [iTC\_CC\_ATP\_SwHA-0068]

[End]

[iTC\_CC\_ATP-SwRS-0195]

如果校准过程中发生下列情况，则从**CALI\_MEASURING**回到**CALI\_WAITING**

If ATP is in the state of **CALI\_MEASURING and** following conditions fulfilled:

* train kinematic has been detected not valid,
* or excessive slip/slide effect has been detected,
* or [WheelMinimumMovement](#WheelMinimumMovement) sign is in the opposite direction of thus observed on first beacon or becomes null,
* or an unexpected beacon has been received. That is, a beacon not belonging calibration measurement couple.

Then, ATP shall abort calibration process and back to **CALI\_WAITING**.

if (([CalibrationState](#CalibrationState)(k-1) = **CALI\_MEASURING**)

and (([ValidTrainKinematic](#ValidTrainKinematic)(k) == **False**)

or ([OdometerState](#OdometerState)(k) is **INVALID**)

or ([SlipSlideDetected](#SlipSlideDetected)(k) == **True**)

or ([CalibrationEnd1RunningForward](#CalibrationEnd1RunningForward)(k-1) != [End1RunningForward](#End1RunningForward)(k))

or ([TrainFilteredStopped](#TrainFilteredStopped)(k) == **True**)

or (([NewBeaconObtained](#NewBeaconObtained)(k) == **True**)

and [TrackMap](#TrackMap).BeaconBelongsToCalibrationCouple([BeaconMessage](#BeaconMessage).Id(k)) == **False**)))

[CalibrationState](#CalibrationState) = **CALI\_WAITING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0145], [iTC\_CC-SyAD-0183]

[End]

#### State “**CALI\_VALIDATING**”

[iTC\_CC\_ATP-SwRS-0196]

当发生以下情况时，认为校准失败，从**CALI\_VALIDATING**回到**CALI\_WAITING**

If ATP is in the state of calibration validation in progress and following conditions fulfilled:

* train kinematic has been detected not valid,
* or excessive slip/slide effect has been detected,
* or [WheelMinimumMovement](#WheelMinimumMovement) sign is in the opposite direction of thus observed on first beacon or becomes null,
* or an unexpected beacon has been received. That is, a beacon not belonging calibration validation couple.

In such case, then ATP shall consider that calibration process as not sable and so back to **CALI\_WAITING**.

if (([CalibrationState](#CalibrationState)(k-1) = **CALI\_VALIDATING**)

and (([ValidTrainKinematic](#ValidTrainKinematic)(k) == **False**)

or ([OdometerState](#OdometerState)(k) is **INVALID**)

or ([SlipSlideDetected](#SlipSlideDetected)(k) == **True**)

or (([CalibrationEnd1RunningForward](#CalibrationEnd1RunningForward)(k-1) != [End1RunningForward](#End1RunningForward)(k))

or ([TrainFilteredStopped](#TrainFilteredStopped)(k) == **True**)

or (([NewBeaconObtained](#NewBeaconObtained)(k) == **True**)

and ([TrackMap](#TrackMap).[IsCalibrationValidationBeacon](#IsCalibrationValidationBeacon)([BeaconMessage](#BeaconMessage).Id(k),

[CalibrationValidationStartBeacon](#CalibrationValidationStartBeacon)(k)) == **False**))))

[CalibrationState](#CalibrationState) = **CALI\_WAITING**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0145]

[End]

[iTC\_CC\_ATP-SwRS-0197]

当读到有效的验证信标，并判断之前测得齿距在有效范围内时，认为校准成功，转入**CALI\_COMPLETED**状态；

否则，校准失败，返回**CALI\_WATING**状态。

If ATP is in the state of calibration validation in progress and following conditions fulfilled:

* a valid beacon has been received and this beacon is one of possible confirmation beacons related to second beacon signaled of calibration measurement,
* and train kinematic was valid,
* and no excessive slip/slide effect was detected,
* and sign of train motion is still identical to thus detected on first beacon signaling,
* and [WheelMinimumMovement](#WheelMinimumMovement) is not null.

Then,

* if resulting calibration range **fully includes** the calibration range in track map, then ATP shall:
* update ATP minimum and maximum calibration with last calibration computed on beacons,
* and shall consider that calibration process is **CALI\_COMPLETED**.
* else: ATP shall consider that calibration process is not usable and so back to **CALI\_WATING** waiting for new measurement calibration beacons.

if ([CalibrationState](#CalibrationState)(k-1) = **CALI\_VALIDATING**)

and ([[NewBeaconObtained](#NewBeaconObtained)](#BeaconMessageReceive)(k) == **True**)

and ([TrackMap](#TrackMap).[IsCalibrationValidationBeacon](#IsCalibrationValidationBeacon)([BeaconMessage](#BeaconMessage).Id(k),

[CalibrationValidationStartBeacon](#CalibrationValidationStartBeacon)(k)))

and ([SlipSlideDetected](#SlipSlideDetected)(k) == **False**)

and ([CalibrationEnd1RunningForward](#CalibrationEnd1RunningForward)(k-1) == [End1RunningForward](#End1RunningForward)(k)))

if (([TrackMap](#TrackMap).[CalibrationCoupleMaxDistance](#CalibrationCoupleMaxDistance)([BeaconMessage](#BeaconMessage).Id(k),

[CalibrationValidationStartBeacon](#CalibrationValidationStartBeacon)(k)) <= [MaxDistanceRanForValidation](#MaxDistanceRanForValidation))

and [TrackMap](#TrackMap).[CalibrationCoupleMinDistance](#CalibrationCoupleMinDistance)([BeaconMessage](#BeaconMessage).Id(k),

[CalibrationValidationStartBeacon](#CalibrationValidationStartBeacon)(k)) >= [MinDistanceRanForValidation](#MinDistanceRanForValidation))

[MinCogCalibration](#MinCogCalibration) = [CalibrationResultMin](#CalibrationResultMin)(k)

[MaxCogCalibration](#MaxCogCalibration) = [CalibrationResultMax](#CalibrationResultMax)(k)

[CalibrationState](#CalibrationState) = **CALI\_COMPLETED**

else:

[CalibrationState](#CalibrationState) = **CALI\_WAITING**

Where:

MaxDistanceRanForValidation

=([abs](#abs)([CalibrationValidationStartPositionMin](#CalibrationValidationStartPositionMin)(k)— [CogPositionAfterTopLoc](#CogPositionAfterTopLoc)(k)) + 1)

\* [CalibrationResultMax](#CalibrationResultMax)(k)

MinDistanceRanForValidation

=([abs](#abs)([CalibrationValidationStartPositionMax](#CalibrationValidationStartPositionMax)(k)— [CogPositionBeforeTopLoc](#CogPositionBeforeTopLoc)(k)) - 1)

\* [CalibrationResultMin](#CalibrationResultMin)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0132], [iTC\_CC-SyAD-0141], [iTC\_CC-SyAD-0142], [iTC\_CC-SyAD-0145], [iTC\_CC\_ATP\_SwHA-0069], [iTC\_CC-SyAD-0183]

[End]

#### State “**CALI\_COMPLETED**”

[iTC\_CC\_ATP-SwRS-0198]

在**CALI\_COMPLETED**状态，最大最小齿距无需再次校准。

If the calibration completed, ATP shall not calibrate when new calibrating beacons read.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0141]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [MaxCogCalibration](#MaxCogCalibration) | √ | √ | NUMERIC\_32 |
| [MinCogCalibration](#MinCogCalibration) | √ | √ | NUMERIC\_32 |

## F3-Locate the Train on Track Map

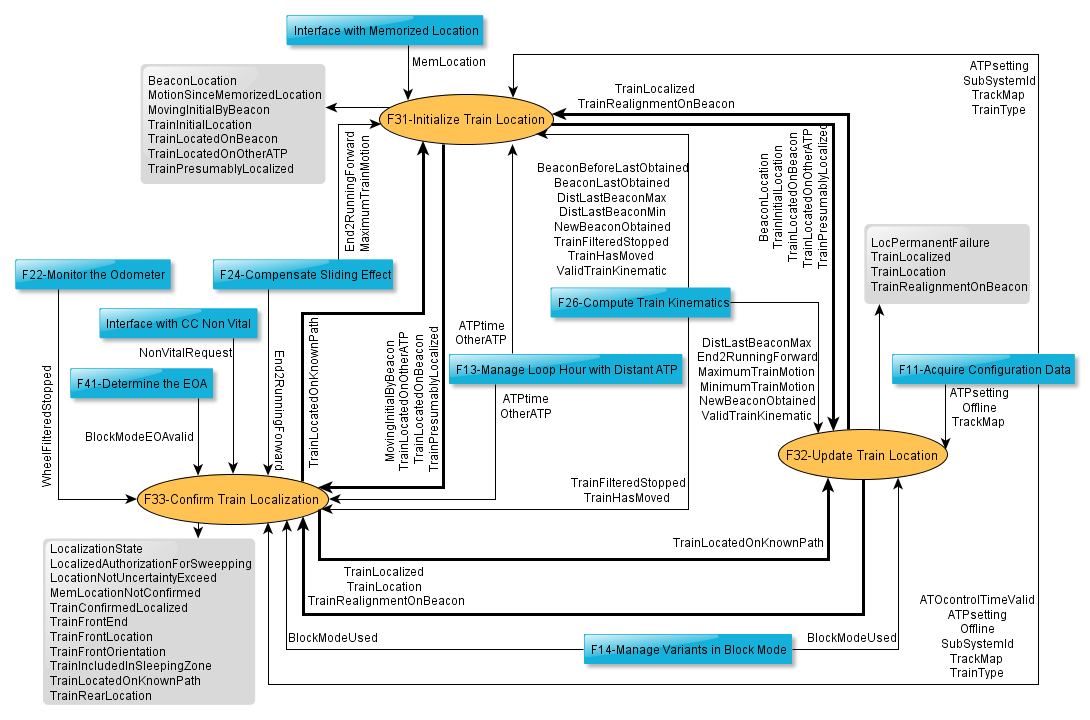


Figure 5‑11 SART modeling of function F3

## F31-Initialize Train Location

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [ATPtime](#ATPtime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [BeaconBeforeLastObtained](#BeaconBeforeLastObtained) | Internal | F27-Compute Train Kinematics |
| [BeaconLastObtained](#BeaconLastObtained) | Internal | F27-Compute Train Kinematics |
| [DistLastBeaconMax](#DistLastBeaconMax) | Internal | F27-Compute Train Kinematics |
| [DistLastBeaconMin](#DistLastBeaconMin) | Internal | F27-Compute Train Kinematics |
| [End2RunningForward](#End2RunningForward) | Internal | F27-Compute Train Kinematics |
| [MaximumTrainMotion](#MaximumTrainMotion) | Internal | F27-Compute Train Kinematics |
| [MemLocation](#MemLocation) | External | Interface with Memorized Location |
| [NewBeaconObtained](#NewBeaconObtained) | Internal | F27-Compute Train Kinematics |
| [OtherATP](#OtherATP) | Internal | F13-Manage Loop Hour with Distant ATP |
| [SubSystemId](#SubSystemId) | Internal | F11-Acquire Configuration Data |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainHasMoved](#TrainHasMoved) | Internal | F27-Compute Train Kinematics |
| [TrainLocalized](#TrainLocalized) | Internal | F32-Update Train Location |
| [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath) | Internal | F33-Confirm Train Localization |
| [TrainRealignmentOnBeacon](#TrainRealignmentOnBeacon) | Internal | F32-Update Train Location |
| [TrainType](#TrainType) | Internal | F11-Acquire Configuration Data |
| [ValidTrainKinematic](#ValidTrainKinematic) | Internal | F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [End2OrientationByBeacon](#End2OrientationByBeacon) | √ | √ | ENUM\_DOT |
| [MemorizedLocationAuthorized](#MemorizedLocationAuthorized) | √ | √ | BOOLEAN |
| [MemorizedLocationAvailable](#MemorizedLocationAvailable) | √ | √ | BOOLEAN |
| [MemorizedLocationEnable](#MemorizedLocationEnable) | √ | √ | BOOLEAN |
| [TrainStoppedStartTime](#TrainStoppedStartTime) | √ | √ | NUMERIC\_32 |

### Processing

ATP可通过三种途径对列车定位进行初始化：

* 读取信标。由于线路上的所有信标位置是存储在线路地图中的，因此当读到信标时，ATP就能知道列车在实际线路中的位置。对于非极性车（即列车车头在线路上的运营方向不定），需要读到连续的2个信标后确认列车位置和实际运营方向；而对于极性车（即指定车头仅能在线路的一个方向运营），读到一个信标即可判断出列车定位。
* 记忆定位。对于允许记忆定位的项目，列车停在线路上指定的休眠区域内时，可记录当时的位置，ATP断电后不易失。当列车被唤醒后，可以通过之前记忆的位置进行定位初始化。为防止ATP断电期间列车移动，在使用记忆定位期间，需限制列车以一个很小的速度在一定范围内行驶，直至读到一个信标确认列车的位置与之前记忆的位置一致。
* 通过远端ATP定位初始化。ATP与另一端车头的ATP之间始终进行定位同步，当两端都判断停车后，失位的一端可使用定位一端的位置进行初始化，以此提高效率。

#### Localization by beacons

[iTC\_CC\_ATP-SwRS-0461]

End2OrientationByBeacon，当定位初始化时，通过经过的信标，判断**END\_2**驾驶室所面对的运营方向。

The orientation of the train END means the **UP** or **DOWN** orientation which this END toward to. When a pair of consecutive beacon read, ATP can determine the orientation for each train END according to the direction of these beacons in track map and the direction of train movement.

def [End2OrientationByBeacon](#End2OrientationByBeacon)(k):

if (Initialization

or not [MovingInitialByBeacon](#MovingInitialByBeacon)(k)):

return **DOT\_UNKNOWN**

elif ([ATPsetting](#ATPsetting).PolarizedTrain):

return [ATPsetting](#ATPsetting).End2Orientation

elif ([NewBeaconObtained](#NewBeaconObtained)(k)

and [TrackMap](#TrackMap).[AreNeighbouredBeacons](#AreNeighbouredBeacons)([BeaconBeforeLastObtained](#BeaconBeforeLastObtained)(k),

[BeaconLastObtained](#BeaconLastObtained)(k))):

if ([End2RunningForward](#End2RunningForward)(k)):

return ([TrackMap](#TrackMap).[OrientationOfNeighbouredBeacons](#OrientationOfNeighbouredBeacons)([BeaconBeforeLastObtained](#BeaconBeforeLastObtained)(k),

[BeaconLastObtained](#BeaconLastObtained)(k)))

else:

return ([TrackMap](#TrackMap).[OrientationOfNeighbouredBeacons](#OrientationOfNeighbouredBeacons)([BeaconLastObtained](#BeaconLastObtained)(k),

[BeaconBeforeLastObtained](#BeaconBeforeLastObtained)(k)))

else:

return [End2OrientationByBeacon](#End2OrientationByBeacon)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0177], [iTC\_CC-SyAD-0180], [iTC\_CC-SyAD-0211], [iTC\_CC-SyAD-1197], [iTC\_CC-SyAD-1198], [iTC\_CC-SyAD-0183], [iTC\_CC\_ATP\_SwHA-0102], [iTC\_CC\_ATP\_SwHA-0240]

[End]



Figure 5‑12 Determine orientation by two neighbor beacons

**NOTES：**

对于极性车，每个车头的运营方向是固定的，如**END\_1**端车头只能向“上行”运营，作为项目配置存储在ATP数据中。

对于对于非极性车，当读到信标时，ATP会根据列车运行方向（即向**END\_2**方向运行或者**END\_1**方向运行）和连续两个信标的“指向”来判断运行方向：

* 对于两信标间没有灯泡线“极点”的情况
* 如果列车向**END\_2**方向运行，而先后读到的信标从A到B为“上行”方向，则**END\_2**端为“上行”，如Figure 5‑12所示；
* 如果列车向**END\_2**方向运行，而先后读到的信标从A到B为“下行”方向，则**END\_2**端为“下行”；
* 如果列车向**END\_1**方向运行，而先后读到的信标从A到B为“上行”方向，则**END\_2**端为“下行”；
* 如果列车向**END\_1**方向运行，而先后读到的信标从A到B为“下行”方向，则**END\_2**端为“上行”；
* 如果相邻两个信标之间有灯泡线的“极点”，则该对信标不能用于方向判别，即无法通过该对信标进行列车定位初始化。

**NOTES：**

所谓灯泡线，是指列车无需折返更换驾驶室，即可完成运营方向由上行切换到下行（或反之）的作业线路。如Figure 5‑13所示，若列车车身跨**UP**极点运行时，则**END\_1**和**END\_2**的方向均为**DOWN**。

On a balloon loop, a train can run to the reverse direction without having to shunt or even stop, refer to Figure 5‑13,which means there are inconsistent orientations for two train ENDs (both **END\_1** and **END\_2** are toward to **UP** or **DOWN**).



Figure 5‑13 The balloon loop

[iTC\_CC\_ATP-SwRS-0248]

BeaconLocation，如果本周期读到了重定位信标（无论是否已经在定位状态），则ATP需根据该信标在线路地图中的坐标计算读到信标时刻**END\_2**车头的位置：

If a valid beacon read, ATP shall calculate actual maximum and minimum location of the **END\_2** according to the beacon location in track map, the distance from beacon antenna to the **END\_2**, the distance after top-loc, the orientation of **END\_2** and the direction of train movement. The beacon location indicate the actual external location of the **END\_2**.

def [BeaconLocation](#BeaconLocation)(k):

if ([NewBeaconObtained](#NewBeaconObtained)(k)):

if ([End2OrientationByBeacon](#End2OrientationByBeacon)(k) is not **None**):

[BeaconLocation](#BeaconLocation).Ext2 = ([TrackMap.LocationUpdateExt2](#LocationUpdateExt2)

([End2RunningForward](#End2RunningForward)(k),

[End2OrientationByBeacon](#End2OrientationByBeacon)(k),

[BeaconLastObtained](#BeaconLastObtained).Location(k),

[MaxMotionOfEnd2](#MaxMotionOfEnd2)(k),

[MinMotionOfEnd2](#MinMotionOfEnd2)(k)))

elif ([TrainLocalized](#TrainLocalized)(k-1)):

[BeaconLocation](#BeaconLocation).Ext2 = ([TrackMap.LocationUpdateExt2](#LocationUpdateExt2)

([End2RunningForward](#End2RunningForward)(k),

[TrainLocation](#TrainLocation)(k-1).Ext2.Ort,

[BeaconLastObtained](#BeaconLastObtained).Location(k),

[MaxMotionOfEnd2](#MaxMotionOfEnd2)(k),

[MinMotionOfEnd2](#MinMotionOfEnd2)(k)))

else:

[BeaconLocation](#BeaconLocation).Ext2 = **None**

[BeaconLocation](#BeaconLocation).Uncertainty = (2 \* [BeaconLastObtained](#BeaconLastObtained)(k).PositionTolerance

+ [abs](#abs)([DistLastBeaconMax](#DistLastBeaconMax)(k) - [DistLastBeaconMin](#DistLastBeaconMin)(k)))

[BeaconLocation](#BeaconLocation).Int2 = [UpdateInt2FromExt2](#UpdateInt2FromExt2)

[BeaconLocation](#BeaconLocation).Ext1 = [UpdateExt1FromExt2](#UpdateExt1FromExt2)

[BeaconLocation](#BeaconLocation).Int1 = [UpdateInt1FromExt2](#UpdateInt1FromExt2)

else:

[BeaconLocation](#BeaconLocation) = **None**

return [BeaconLocation](#BeaconLocation)

其中，所用到的内部函数分别为：

def MaxMotionOfEnd2(k)

return ([DistLastBeaconMax](#DistLastBeaconMax)(k)

- [ATPsetting](#ATPsetting).CCcoreEnd2BeaconAntennaDistance[[CoreId](#CoreId)(k)]

- [BeaconLastObtained](#BeaconLastObtained)(k).PositionTolerance)

def MinMotionOfEnd2(k)

return ([DistLastBeaconMin](#DistLastBeaconMin)(k)

- [ATPsetting](#ATPsetting).CCcoreEnd2BeaconAntennaDistance[[CoreId](#CoreId)(k)]

- [BeaconLastObtained](#BeaconLastObtained)(k).PositionTolerance)

UpdateInt2FromExt2，根据Figure 5‑14，通过Ext2定位计算End2端的内侧定位Int2。如果计算范围内有非受控道岔或者轨道尽头，则Int2定位应设置为**None**。如果Int2与Ext2之间有灯泡线极点，则二者的方向将不同。

UpdateExt1FromExt2，根据Figure 5‑14，通过Ext2定位计算End1端的外侧定位Ext1。如果计算范围内有非受控道岔或者轨道尽头，则Ext1定位应设置为**None**。如果Ext1与Ext2之间有灯泡线极点，则二者的方向将相同。

UpdateInt1FromExt2，根据Figure 5‑14，通过Ext2定位计算End1端的内侧定位Int1。如果计算范围内有非受控道岔或者轨道尽头，则Int1定位应设置为**None**。如果Int1与Ext2之间有灯泡线极点，则二者的方向将相同。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0176], [iTC\_CC-SyAD-0178], [iTC\_CC-SyAD-0180], [iTC\_CC-SyAD-0182], [iTC\_CC-SyAD-0196], [iTC\_CC-SyAD-0197], [iTC\_CC-SyAD-0202], [iTC\_CC\_ATP\_SwHA-0098], [iTC\_CC-SyAD-0183]

[End]

**NOTES:**

如Figure 5‑14所示，表示列车两端车头的内外侧四个定位的关系。

* 其中对于每一端车头的内外侧定位，其坐标相差一个定位误差长度。
* 而对于一端的外侧定位和另一端的内侧定位，其坐标相差配置数据中的列车长度。
* 对于每一个定位，均有其相应的运营方向。注意，如果车身范围内有灯泡线极点，则两端车头的运营方向可能相同。



Figure 5‑14 Train location with orientation

[iTC\_CC\_ATP-SwRS-0276]

MovingInitialByBeacon，是否在信标初始化定位过程中。

def [MovingInitialByBeacon](#MovingInitialByBeacon)(k):

if (Initialization

or [TrainLocalized](#TrainLocalized)(k-1)

or [TrainFilteredStopped](#TrainFilteredStopped)(k)

or ([End1RunningForward](#End1RunningForward)(k) and not [End1RunningForward](#End1RunningForward)(k-1))

or ([End2RunningForward](#End2RunningForward)(k) and not [End2RunningForward](#End2RunningForward)(k-1))

or [abs](#abs)([DistLastBeaconMax](#DistLastBeaconMax)(k)) >= [ATPsetting](#ATPsetting).BeaconPairMaxDistance):

return **False**

elif ([NewBeaconObtained](#NewBeaconObtained)(k)):

return **True**

else:

return [MovingInitialByBeacon](#MovingInitialByBeacon)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source= [iTC\_CC-SyAD-0176], [iTC\_CC-SyAD-0180], [iTC\_CC-SyAD-0183]

[End]

[iTC\_CC\_ATP-SwRS-0280]

TrainLocatedOnBeacon，列车通过信标进行初始化定位（该值仅在处理信标的周期为**True**）。如果在定位初始化阶段读到信标，且能够根据该信标的位置计算出列车的定位，（即车身范围内没有轨道边界或未知状态的道岔），即认为初始化定位成功。

def [TrainLocatedOnBeacon](#TrainLocatedOnBeacon)(k):

return ([MovingInitialByBeacon](#MovingInitialByBeacon)(k)

and [NewBeaconObtained](#NewBeaconObtained)(k)

and ([ATPsetting](#ATPsetting).PolarizedTrain

or ([MovingInitialByBeacon](#MovingInitialByBeacon)(k-1)

and [TrackMap](#TrackMap).[AreNeighbouredBeacons](#AreNeighbouredBeacons)([BeaconBeforeLastObtained](#BeaconBeforeLastObtained)(k),

[BeaconLastObtained](#BeaconLastObtained)(k))))

and [BeaconLocation](#BeaconLocation)(k) is not **None**)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source= [iTC\_CC-SyAD-0176], [iTC\_CC-SyAD-0177], [iTC\_CC-SyAD-0180], [iTC\_CC-SyAD-0183], [iTC\_CC-SyAD-1197], [iTC\_CC-SyAD-1198], [iTC\_CC\_ATP\_SwHA-0101]

[End]

#### Localization by memorized location

[iTC\_CC\_ATP-SwRS-0597]

MemorizedLocationAuthorized，项目配置是否授权使用记忆定位

def [MemorizedLocationAuthorized](#MemorizedLocationAuthorized)(k):

return [Offline.GetMemorizedLocationAuthorized](#GetMemorizedLocationAuthorized)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1206]

[End]

[iTC\_CC\_ATP-SwRS-0643]

MemorizedLocationEnable，仅在刚上电车还未动时允许使用记忆定位

def [MemorizedLocationEnable](#MemorizedLocationEnable)(k):

return (not [TrainHasMoved](#TrainHasMoved)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1207]

[End]

[iTC\_CC\_ATP-SwRS-0644]

MemorizedLocationAvailable，记忆定位是否可用

def [MemorizedLocationAvailable](#MemorizedLocationAvailable)(k):

return ([TrainFilteredStopped](#TrainFilteredStopped)(k)

and [ValidTrainKinematic](#ValidTrainKinematic)(k)

and [MemorizedLocationAuthorized](#MemorizedLocationAuthorized)(k)

and [MemorizedLocationEnable](#MemorizedLocationEnable)(k)

and Message.[VitalChecksumValid](#VitalChecksumValid)([MemLocation](#MemLocation))

and [MemLocation](#MemLocation).MemLocVersion == **MEM\_LOCATION\_VERSION**

and [MemLocation](#MemLocation).TrainType == [TrainType](#TrainType)(k)

and [MemLocation](#MemLocation).TrainId == [SubSystemId](#SubSystemId)(k)

and [MemLocation](#MemLocation).SleepAreaId == ([TrackMap](#TrackMap).[ExistZoneLocationIncluded](#ExistZoneLocationIncluded)

(**SGL\_SLEEPING\_ZONE**, [MemLocation](#MemLocation)(k).Ext2).Id)

and [MemLocation](#MemLocation).SleepAreaVersion == ([TrackMap](#TrackMap).[ExistZoneLocationIncluded](#ExistZoneLocationIncluded)

(**SGL\_SLEEPING\_ZONE**, [MemLocation](#MemLocation)(k).Ext2).Version)

and [MemLocation](#MemLocation).TrainLength == [ATPsetting](#ATPsetting).LocationTrainLength)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1202], [iTC\_CC-SyAD-1203], [iTC\_CC-SyAD-1205], [iTC\_CC-SyAD-1207], [iTC\_CC\_ATP\_SwHA-0241], [iTC\_CC\_ATP\_SwHA-0242]

[End]

[iTC\_CC\_ATP-SwRS-0645]

MotionSinceMemorizedLocation，记录自唤醒后运行了多少距离

def [MotionSinceMemorizedLocation](#MotionSinceMemorizedLocation)(k):

if ([TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP)(k-1)):

return [OtherATP](#OtherATP)(k).MotionSinceMemLoc

elif ([MemorizedLocationAvailable](#MemorizedLocationAvailable)(k)

or not [MemLocationNotConfirmed](#MemLocationNotConfirmed)(k-1)):

return 0

else:

return ([MotionSinceMemorizedLocation](#MotionSinceMemorizedLocation)(k-1) + [MaximumTrainMotion](#MaximumTrainMotion)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1208], [iTC\_CC-SyAD-1209]

[End]

[iTC\_CC\_ATP-SwRS-0646]

TrainPresumablyLocalized，列车使用记忆定位，但还未读到确认信标的状态。待已经通过信标重定位，或者失位后，清除该值。

def [TrainPresumablyLocalized](#TrainPresumablyLocalized)(k):

if ([MemorizedLocationAvailable](#MemorizedLocationAvailable)(k)

and not [TrainPresumablyLocalized](#TrainPresumablyLocalized)(k-1)):

return **True**

elif ([TrainPresumablyLocalized](#TrainPresumablyLocalized)(k-1)

and (not [TrainLocalized](#TrainLocalized)(k-1)

or [TrainRealignmentOnBeacon](#TrainRealignmentOnBeacon)(k-1)):

return **False**

else:

return [TrainPresumablyLocalized](#TrainPresumablyLocalized)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1207], [iTC\_CC-SyAD-1208], [iTC\_CC-SyAD-1209]

[End]

#### Localization by distant ATP location

[iTC\_CC\_ATP-SwRS-0647]

TrainStoppedStartTime，记录开始停车的时间

def [TrainStoppedStartTime](#TrainStoppedStartTime)(k):

if (Initialization

or (not [TrainFilteredStopped](#TrainFilteredStopped)(k-1)

and [TrainFilteredStopped](#TrainFilteredStopped)(k)):

return [ATPtime](#ATPtime)(k)

else:

return [TrainStoppedStartTime](#TrainStoppedStartTime)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1211]

[End]

[iTC\_CC\_ATP-SwRS-0648]

TrainLocatedOnOtherATP，本端和远端都在停车状态时，才有可能使用远端定位

Only when ATP and redundant ATP are all in filtered stopped state, can ATP use redundant ATP location for initialization.

def [TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP)(k):

if ([OtherATP](#OtherATP)(k).LocatedOnKnownPath

and [TrainFilteredStopped](#TrainFilteredStopped)(k)

and [OtherATP](#OtherATP)(k).TrainFilteredStopped

and [Message.IsMoreRecent](#IsMoreRecent)([OtherATP](#OtherATP)(k).LatestTimeOtherCore, [TrainStoppedStartTime](#TrainStoppedStartTime)(k))

and not [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k-1)

and not [TrainPresumablyLocalized](#TrainPresumablyLocalized)(k)):

[TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP) = **True**

else:

[TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP) = **False**

return [TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1211], [iTC\_CC\_ATP\_SwHA-0244]

[End]

#### Initialize train location

[iTC\_CC\_ATP-SwRS-0649]

TrainInitialLocation，记录列车通过远端ATP、记忆定位、或信标初始化时的位置。

* 如果列车失位，则清除该位置
* 如果列车保持定位，则保留该位置

ATP determine the initial train location by redundant ATP, memorized location and beacon location in order. If train delocalized, the train location should be clear.

def [TrainInitialLocation](#TrainInitialLocation)(k):

if ([TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP)(k)):

return [OtherATP](#OtherATP)(k).Location

elif ([TrainPresumablyLocalized](#TrainPresumablyLocalized)(k)

and not [TrainHasMoved](#TrainHasMoved)(k)):

return [MemLocation](#MemLocation)(k)

elif ([TrainLocatedOnBeacon](#TrainLocatedOnBeacon)(k)):

return [BeaconLocation](#BeaconLocation)(k)

elif ([**TrainLocalized**](#TrainLocalized)(k-1)):

return [TrainInitialLocation](#TrainInitialLocation)(k-1)

else:

return **None**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1207], [iTC\_CC-SyAD-1212], [iTC\_CC\_ATP\_SwHA-0245]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [BeaconLocation](#BeaconLocation) | √ | √ | ST\_LOCATION\_TRAIN |
| [MotionSinceMemorizedLocation](#MotionSinceMemorizedLocation) | √ | √ | NUMERIC\_32 |
| [MovingInitialByBeacon](#MovingInitialByBeacon) | √ | √ | BOOLEAN |
| [TrainInitialLocation](#TrainInitialLocation) | √ | √ | ST\_LOCATION\_TRAIN |
| [TrainLocatedOnBeacon](#TrainLocatedOnBeacon) | √ | √ | BOOLEAN |
| [TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP) | √ | √ | BOOLEAN |
| [TrainPresumablyLocalized](#TrainPresumablyLocalized) | √ | √ | BOOLEAN |

## F32-Update Train Location

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [BeaconLocation](#BeaconLocation) | Internal | F31-Initialize Train Location |
| [BlockModeUsed](#BlockModeUsed) | Internal | F14-Manage Variants in Block Mode |
| [DistLastBeaconMax](#DistLastBeaconMax) | Internal | F27-Compute Train Kinematics |
| [End2RunningForward](#End2RunningForward) | Internal | F27-Compute Train Kinematics |
| [MaximumTrainMotion](#MaximumTrainMotion) | Internal | F27-Compute Train Kinematics |
| [MinimumTrainMotion](#MinimumTrainMotion) | Internal | F27-Compute Train Kinematics |
| [NewBeaconObtained](#NewBeaconObtained) | Internal | F27-Compute Train Kinematics |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [OtherATP](#OtherATP) | Internal | F13-Manage Loop Hour with Distant ATP |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainInitialLocation](#TrainInitialLocation) | Internal | F31-Initialize Train Location |
| [TrainLocatedOnBeacon](#TrainLocatedOnBeacon) | Internal | F31-Initialize Train Location |
| [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath) | Internal | F33-Confirm Train Localization |
| [TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP) | Internal | F31-Initialize Train Location |
| [TrainPresumablyLocalized](#TrainPresumablyLocalized) | Internal | F31-Initialize Train Location |
| [ValidTrainKinematic](#ValidTrainKinematic) | Internal | F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [CBTCwithoutKnownPath](#CBTCwithoutKnownPath) | √ | √ | BOOLEAN |
| [InverseLocation](#InverseLocation) | √ | √ | BOOLEAN |
| [LocalizationFault](#LocalizationFault) | √ | √ | BOOLEAN |
| [LocationAfterReloc](#LocationAfterReloc) | √ | √ | ST\_LOCATION\_TRAIN |
| [LocationBeforeReloc](#LocationBeforeReloc) | √ | √ | ST\_LOCATION\_TRAIN |
| [LocationUncertaintyExceed](#LocationUncertaintyExceed) | √ | √ | BOOLEAN |
| [LocationUntravelable](#LocationUntravelable) | √ | √ | BOOLEAN |
| [MotionSinceLastReloc](#MotionSinceLastReloc) | √ | √ | NUMERIC\_32 |
| [RealignmentFailed](#RealignmentFailed) | √ | √ | BOOLEAN |
| [TrainUnitIntegrity](#TrainUnitIntegrity) | √ | √ | BOOLEAN |

### Processing

#### Location calculating by movement

[iTC\_CC\_ATP-SwRS-0259]

LocationBeforeReloc，上周期列车已定位的情况下，使用里程计测得的位移来更新列车定位。

If train has localized on the track map, according to the orientation of **END\_2**, ATP using the maximum and minimum train motion to update the external or internal location of the **END\_2**.

def [LocationBeforeReloc](#LocationBeforeReloc)(k):

if ([TrainLocalized](#TrainLocalized)(k-1) and [ValidTrainKinematic](#ValidTrainKinematic)(k)):

if ([OdometerState](#OdometerState)(k) is **INITIALIZED**):

if ([End2RunningForward](#End2RunningForward)(k)):

[LocationBeforeReloc](#LocationBeforeReloc).Uncertainty = ([TrainLocation](#TrainLocation)(k-1).Uncertainty

- ([MaximumTrainMotion](#MaximumTrainMotion)(k) - [MinimumTrainMotion](#MinimumTrainMotion)(k)))

else:

[LocationBeforeReloc](#LocationBeforeReloc).Uncertainty = ([TrainLocation](#TrainLocation)(k-1).Uncertainty

+ ([MaximumTrainMotion](#MaximumTrainMotion)(k) - [MinimumTrainMotion](#MinimumTrainMotion)(k)))

else:

[LocationBeforeReloc](#LocationBeforeReloc).Uncertainty = ([TrainLocation](#TrainLocation)(k-1).Uncertainty

+ abs([MaximumTrainMotion](#MaximumTrainMotion)(k))

+ abs([MinimumTrainMotion](#MinimumTrainMotion)(k)))

[LocationBeforeReloc](#LocationBeforeReloc).Ext2 = ([TrackMap.LocationUpdateExt2](#LocationUpdateExt2)([End2RunningForward](#End2RunningForward)(k),

[TrainLocation](#TrainLocation)(k-1).Ext2.Ort,

[TrainLocation](#TrainLocation)(k-1).Ext2,

[MaximumTrainMotion](#MaximumTrainMotion)(k),

[MinimumTrainMotion](#MinimumTrainMotion)(k)))

[LocationBeforeReloc](#LocationBeforeReloc).Int2 = [UpdateInt2FromExt2](#UpdateInt2FromExt2)

[LocationBeforeReloc](#LocationBeforeReloc).Ext1 = [UpdateExt1FromExt2](#UpdateExt1FromExt2)

[LocationBeforeReloc](#LocationBeforeReloc).Int1 = [UpdateInt1FromExt2](#UpdateInt1FromExt2)

else:

[LocationBeforeReloc](#LocationBeforeReloc) = **None**

return [LocationBeforeReloc](#LocationBeforeReloc)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0172], [iTC\_CC-SyAD-0184], [iTC\_CC\_ATP\_SwHA-0096]

[End]

#### Delocalization detecting

[iTC\_CC\_ATP-SwRS-0254]

LocationUntravelable，判断车身范围内是否有线路边界或者状态不符的道岔

* 当上周期列车定位，并满足以下条件时，本周期设置为**True**。
* 如果车尾最小定位到车头最大定位之间存在状态未知的道岔(包括发散或汇聚节点)；
* 或者，如果车尾最小定位到车头最大定位之间存在变量状态与之前列车位置不符的发散汇聚节点；
* 或者，轨道边界在列车定位范围内；
* 否则，设置该值为**False**。

ATP shall determine whether there is an unknown-status point intersecting with the train location.

* If the train has localized at the previous cycle, and fulfills one of the following conditions:
* There is an unknown-status divergence of convergence located in the range from train tail to train head, then ATP shall set as **True**;
* Or else:, if there is a convergence with reverse route located in the range from train tail to train head, the ATP shall set as **True**;
* Or else, train crossed the boundary of ATC area, shall set as **True**.
* Otherwise, set as **False**.

def [LocationUntravelable](#LocationUntravelable)(k):

return ([TrainLocalized](#TrainLocalized)(k-1)

and (not ([TrackMap](#TrackMap).ReachableBetweenTwoLocations

([LocationBeforeReloc](#LocationBeforeReloc)(k).Ext2, [LocationBeforeReloc](#LocationBeforeReloc)(k).Ext1,

([abs](#abs)([MaximumTrainMotion](#MaximumTrainMotion)) + [ATPsetting](#ATPsetting).LocationTrainLength

+ [LocationBeforeReloc](#LocationBeforeReloc)(k).Uncertainty)))

or not ([TrackMap](#TrackMap).ReachableBetweenTwoLocations

([LocationBeforeReloc](#LocationBeforeReloc)(k).Ext1, [LocationBeforeReloc](#LocationBeforeReloc)(k).Ext2,

([abs](#abs)([MaximumTrainMotion](#MaximumTrainMotion)) + [ATPsetting](#ATPsetting).LocationTrainLength

+ [LocationBeforeReloc](#LocationBeforeReloc)(k).Uncertainty)))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0190], [iTC\_CC-SyAD-0191], [iTC\_CC-SyAD-1205], [iTC\_CC\_ATP\_SwHA-0100]

[End]

[iTC\_CC\_ATP-SwRS-0460]

InverseLocation，判断**END\_2**车头的外侧和内侧定位顺序是否正确。

ATP shall determine the correct order of the external and internal location of train END.

def [InverseLocation](#InverseLocation)(k):

return ([LocationBeforeReloc](#LocationBeforeReloc)(k).Uncertainty < 0)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0188]

[End]

[iTC\_CC\_ATP-SwRS-0253]

LocationUncertaintyExceed，列车定位状态下，每周期计算列车外侧定位和内侧之间的距离是否超过最大允许误差。

ATP shall calculate the uncertain distance between the external and internal locations of train **END\_2**.

def [LocationUncertaintyExceed](#LocationUncertaintyExceed)(k):

return ([LocationBeforeReloc](#LocationBeforeReloc)(k).Uncertainty > [ATPsetting](#ATPsetting).LocationMaxUncertaintyConfirmed)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0189]

[End]

[iTC\_CC\_ATP-SwRS-0650]

TrainRealignmentOnBeacon，是否在信标上重定位成功

def [TrainRealignmentOnBeacon](#TrainRealignmentOnBeacon)(k):

return ([TrainLocalized](#TrainLocalized)(k-1)

and [NewBeaconObtained](#NewBeaconObtained)(k)

and ([TrackMap](#TrackMap).[IntersectionOfTwoZones](#IntersectionOfTwoZones)([BeaconLocation](#BeaconLocation).Int2, [BeaconLocation](#BeaconLocation).Ext2,

[LocationBeforeReloc](#LocationBeforeReloc).Int2, [LocationBeforeReloc](#LocationBeforeReloc).Ext2)

is not **None**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0202], [iTC\_CC-SyAD-1208], [iTC\_CC\_ATP\_SwHA-0099]

[End]

[iTC\_CC\_ATP-SwRS-0768]

RealignmentFailed，上周期定位状态下，如果读到新的重定位信标，但ATP根据位移计算的最大最小定位，与通过读到信标位置计算的最大最小定位之间没有交集，则认为重定位失败。

def [RealignmentFailed](#RealignmentFailed)(k):

return ([TrainLocalized](#TrainLocalized)(k-1)

and [NewBeaconObtained](#NewBeaconObtained)(k)

and ([TrackMap](#TrackMap).[IntersectionOfTwoZones](#IntersectionOfTwoZones)([BeaconLocation](#BeaconLocation).Int2, [BeaconLocation](#BeaconLocation).Ext2,

[LocationBeforeReloc](#LocationBeforeReloc).Int2, [LocationBeforeReloc](#LocationBeforeReloc).Ext2)

is **None**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0176], [iTC\_CC-SyAD-0196], [iTC\_CC-SyAD-0197], [iTC\_CC-SyAD-0198], [iTC\_CC-SyAD-0199], [iTC\_CC\_ATP\_SwHA-0099], [iTC\_CC-SyAD-0183]

[End]

[iTC\_CC\_ATP-SwRS-0745]

LocationAfterReloc，经过信标重定位后的列车定位

def [LocationAfterReloc](#LocationAfterReloc)(k):

if ([TrainRealignmentOnBeacon](#TrainRealignmentOnBeacon)(k)):

([LocationAfterReloc](#LocationAfterReloc).Int2,

[LocationAfterReloc](#LocationAfterReloc).Ext2) = ([TrackMap](#TrackMap).[IntersectionOfTwoZones](#IntersectionOfTwoZones)([LocationBeforeReloc](#LocationBeforeReloc)(k).Int2,

[LocationBeforeReloc](#LocationBeforeReloc)(k).Ext2,

[BeaconLocation](#BeaconLocation)(k).Int2,

[BeaconLocation](#BeaconLocation)(k).Ext2))

[LocationAfterReloc](#LocationAfterReloc).Uncertainty = ([TrackMap](#TrackMap).[DistanceBtwTwoLocs](#DistanceBtwTwoLocs)([LocationAfterReloc](#LocationAfterReloc).Int2,

[LocationAfterReloc](#LocationAfterReloc).Ext2,

[ATPsetting](#ATPsetting).LocationMaxUncertaintyConfirmed))

[LocationAfterReloc](#LocationAfterReloc).Int1 = [UpdateInt1FromExt2](#UpdateInt1FromExt2)

[LocationAfterReloc](#LocationAfterReloc).Ext1 = [UpdateExt1FromExt2](#UpdateExt1FromExt2)

elif (not [TrainLocalized](#TrainLocalized)(k-1)):

[LocationAfterReloc](#LocationAfterReloc) = **None**

else:

[LocationAfterReloc](#LocationAfterReloc) = [LocationAfterReloc](#LocationAfterReloc)(k-1)

return [LocationAfterReloc](#LocationAfterReloc)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0176], [iTC\_CC-SyAD-0196], [iTC\_CC\_ATP\_SwHA-0099]

[End]

[iTC\_CC\_ATP-SwRS-0250]

LocPermanentFailure，在列车已定位，且未使用非确认的BM变量情况下，若发生重定位失败，则永久失位。

def [LocPermanentFailure](#LocPermanentFailure)(k):

return ([LocPermanentFailure](#LocPermanentFailure)(k-1)

or ([TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k-1)

and [RealignmentFailed](#RealignmentFailed)(k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0176], [iTC\_CC-SyAD-0196], [iTC\_CC-SyAD-0197], [iTC\_CC-SyAD-0198], [iTC\_CC-SyAD-0199], [iTC\_CC\_ATP\_SwHA-0099], [iTC\_CC-SyAD-0183]

[End]

[iTC\_CC\_ATP-SwRS-0651]

MotionSinceLastReloc，记录自上次信标重定位后的运行距离绝对值

def [MotionSinceLastReloc](#MotionSinceLastReloc)(k):

if ([TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP)(k)):

return [OtherATP](#OtherATP)(k).[MotionSinceLastReloc](#MotionSinceLastReloc)

elif ([TrainRealignmentOnBeacon](#TrainRealignmentOnBeacon)(k)):

return [abs](#abs)([DistLastBeaconMax](#DistLastBeaconMax)(k))

elif (not [TrainLocalized](#TrainLocalized)(k-1)):

return 0

else:

return ([MotionSinceLastReloc](#MotionSinceLastReloc)(k-1) + [abs](#abs)([MaximumTrainMotion](#MaximumTrainMotion)(k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0187]

[End]

[iTC\_CC\_ATP-SwRS-0652]

CBTCwithoutKnownPath，在CBTC下，若上周期定位path为假，则失位

def [CBTCwithoutKnownPath](#CBTCwithoutKnownPath)(k):

return ([TrainLocalized](#TrainLocalized)(k-1)

and not [LocationPathKnown](#LocationPathKnown)(k-1)

and not [BlockModeUsed](#BlockModeUsed)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1213]

[End]

[iTC\_CC\_ATP-SwRS-0074]

TrainUnitIntegrity，任一端车头能保证列车完整性，则认为车辆完整性能被保证。如果该项目未配置列车完整性采集，则认为列车完整性已由车辆保证。其状态来自于项目可配置的列车输入采集。

If either of ends can ensure the train integrity, ATP shall set [TrainUnitIntegrity](#TrainUnitIntegrity) as **True**. If the project is not configured with the capture of train integrity, it is sure that the train can guarantees the integrity.

def [TrainUnitIntegrity](#TrainUnitIntegrity)(k):

return [Offline](#Offline).[GetTrainUnitIntegrity](#GetTrainUnitIntegrity)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0195], [iTC\_CC-SyAD-0334], [iTC\_CC-SyAD-0335], [iTC\_CC-SyAD-0336], [iTC\_CC-SyAD-1003], [iTC\_CC\_ATP\_SwHA-0208]

[End]

[iTC\_CC\_ATP-SwRS-0278]

LocalizationFault用于表明是否发生定位错误。判断条件如下:

If the localization state is not **NOT\_LOCALIZED**, ATP shall determine whether the localization fault happens or not, according to the following pseudo-codes:

def [LocalizationFault](#LocalizationFault)(k):

return (not [TrainUnitIntegrity](#TrainUnitIntegrity)(k)

or not [ValidTrainKinematic](#ValidTrainKinematic)(k)

or [LocationUntravelable](#LocationUntravelable)(k)

or [InverseLocation](#InverseLocation)(k)

or [LocationUncertaintyExceed](#LocationUncertaintyExceed)(k)

or [RealignmentFailed](#RealignmentFailed)(k)

or [LocPermanentFailure](#LocPermanentFailure)(k)

or ([MotionSinceLastReloc](#MotionSinceLastReloc)(k) > [ATPsetting](#ATPsetting).LocationBeaconValidityDistance)

or ([MemLocationNotConfirmed](#MemLocationNotConfirmed)(k-1)

and [abs](#abs)([MotionSinceMemorizedLocation](#MotionSinceMemorizedLocation)(k)) >= [ATPsetting](#ATPsetting).MemLocValidityDistance)

or [CBTCwithoutKnownPath](#CBTCwithoutKnownPath)(k)

or [CoupledTypeInconsistent](#CoupledTypeInconsistent)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source= [iTC\_CC-SyAD-0186], [iTC\_CC-SyAD-0187], [iTC\_CC-SyAD-0188], [iTC\_CC-SyAD-0189], [iTC\_CC-SyAD-0190], [iTC\_CC-SyAD-0191], [iTC\_CC-SyAD-0192], [iTC\_CC-SyAD-0193], [iTC\_CC-SyAD-0195], [iTC\_CC-SyAD-0199], [iTC\_CC-SyAD-0364], [iTC\_CC-SyAD-0137], [iTC\_CC-SyAD-1209], [iTC\_CC\_ATP\_SwHA-0194]

[End]

#### Train location calculating

[iTC\_CC\_ATP-SwRS-0283]

TrainLocalized，表示当前列车是否定位。

Only the localization state is **LOCALIZED**, ATP shall consider the train has localized.

def [TrainLocalized](#TrainLocalized)(k):

if (Initialization

or [LocalizationFault](#LocalizationFault)(k)):

return **False**

elif (not [TrainLocalized](#TrainLocalized)(k-1)

and ([TrainPresumablyLocalized](#TrainPresumablyLocalized)(k)

or [TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP)(k)

or [TrainLocatedOnBeacon](#TrainLocatedOnBeacon)(k))):

return **True**

else:

return [TrainLocalized](#TrainLocalized)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0213], [iTC\_CC-SyAD-0137]

[End]

当列车定位初始化后，ATP可根据里程计测得并经打滑补偿和轴锁判断处理的列车位移，每周期更新列车在线路地图中的位置。如果再读到信标，则ATP可根据该信标的位置对之前的定位进行重新校正。考虑到安全，ATP需维护列车每端车头的外侧和内侧两组定位信息。

When the train passed the continuous two beacons, ATP can judge the initial location and direction according to the position and the sequences of above-mentioned beacons in track map. Later, ATP can update the train location in the track map in each cycle based on the train movement combined with sliding overestimation and wheel block consideration. If ATP received a new beacon, it will realign the train location according to this beacon. For safety, ATP needs to maintain the location information from the external and internal side of each train

end.

[iTC\_CC\_ATP-SwRS-0258]

TrainLocation，列车End1和End2端定位。

分为以下四种情况：

* 本周期非定位；
* 本周期刚初始化；
* 本周期经过信标重定位；
* 本周期使用位移累加定位。

def [TrainLocation](#TrainLocation)(k):

if (not [TrainLocalized](#TrainLocalized)(k)):

return **None**

elif (not [TrainLocalized](#TrainLocalized)(k-1)):

return [TrainInitialLocation](#TrainInitialLocation)(k)

elif ([TrainRealignmentOnBeacon](#TrainRealignmentOnBeacon)(k)):

return [LocationAfterReloc](#LocationAfterReloc)(k)

else:

return [LocationBeforeReloc](#LocationBeforeReloc)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0172], [iTC\_CC-SyAD-0176], [iTC\_CC-SyAD-0185], [iTC\_CC-SyAD-0196], [iTC\_CC\_ATP\_SwHA-0099]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [LocPermanentFailure](#LocPermanentFailure) | √ | √ | BOOLEAN |
| [TrainLocalized](#TrainLocalized) | √ | √ | BOOLEAN |
| [TrainLocation](#TrainLocation) | √ | √ | ST\_LOCATION\_TRAIN |
| [TrainRealignmentOnBeacon](#TrainRealignmentOnBeacon) | √ | √ | BOOLEAN |

## F33-Confirm Train Localization

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATOcontrolTimeValid](#ATOcontrolTimeValid) | Internal | F11-Acquire Configuration Data |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [ATPtime](#ATPtime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [BlockModeEOAvalid](#BlockModeEOAvalid) | Internal | F41-Determine the EOA |
| [BlockModeUsed](#BlockModeUsed) | Internal | F14-Manage Variants in Block Mode |
| [End2RunningForward](#End2RunningForward) | Internal | F27-Compute Train Kinematics |
| [MovingInitialByBeacon](#MovingInitialByBeacon) | Internal | F31-Initialize Train Location |
| [NonVitalRequest](#NonVitalRequest) | External | Interface with CC Non Vital |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [OtherATP](#OtherATP) | Internal | F13-Manage Loop Hour with Distant ATP |
| [SubSystemId](#SubSystemId) | Internal | F11-Acquire Configuration Data |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainHasMoved](#TrainHasMoved) | Internal | F27-Compute Train Kinematics |
| [TrainLocalized](#TrainLocalized) | Internal | F32-Update Train Location |
| [TrainLocatedOnBeacon](#TrainLocatedOnBeacon) | Internal | F31-Initialize Train Location |
| [TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP) | Internal | F31-Initialize Train Location |
| [TrainLocation](#TrainLocation) | Internal | F32-Update Train Location |
| [TrainPresumablyLocalized](#TrainPresumablyLocalized) | Internal | F31-Initialize Train Location |
| [TrainRealignmentOnBeacon](#TrainRealignmentOnBeacon) | Internal | F32-Update Train Location |
| [TrainType](#TrainType) | Internal | F11-Acquire Configuration Data |
| [WheelFilteredStopped](#WheelFilteredStopped) | Internal | F22-Monitor the Odometer |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [ClearingMemLocRequest](#ClearingMemLocRequest) | √ | √ | BOOLEAN |
| [DriverInCab\_1](#DriverInCab_1) | √ | √ | BOOLEAN |
| [DriverInCab\_2](#DriverInCab_2) | √ | √ | BOOLEAN |
| [DriverInTrain](#DriverInTrain) | √ | √ | BOOLEAN |
| [LocationPathKnown](#LocationPathKnown) | √ | √ | BOOLEAN |
| [LocationUncertaintyExceedTime](#LocationUncertaintyExceedTime) | √ | √ | BOOLEAN |
| [MemLocWritten](#MemLocWritten) | √ | √ | BOOLEAN |
| [NonVitalSelectedFrontEnd](#NonVitalSelectedFrontEnd) | × | √ | ENUM\_CAB\_ID |
| [WritingMemLocRequest](#WritingMemLocRequest) | √ | √ | BOOLEAN |

### Processing

#### Determine the train front

ATP根据项目配置参数，在指定端口获取司机激活的驾驶室状态，据此来判断车头方向。如果司机未激活任意一端驾驶室，则ATP应当根据CCNV送来的驾驶室进行车头判断。

ATP shall determine the train front by cab activation from project configured VIOM input port. If the driver did not activate either end of the cab, ATP shall judge the train front according to the info from CCNV.

[iTC\_CC\_ATP-SwRS-0076]

DriverInCab\_1或DriverInCab\_2，如果采集到某端的驾驶室被激活，则ATP认为司机在该端驾驶室。其状态来自于项目可配置的列车输入采集。

ATP shall consider the driver is in this cab if it captures that either end of cab activated, which shown by the data from [DriverInCab\_1](#DriverInCab_1) or [DriverInCab\_2](#DriverInCab_2).

def [DriverInCab\_1](#DriverInCab_1)(k):

return [Offline](#Offline).[GetDriverInCab\_1](#GetDriverInCab_1)(k)

def [DriverInCab\_2](#DriverInCab_2)(k):

return [Offline](#Offline).[GetDriverInCab\_2](#GetDriverInCab_2)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0346], [iTC\_CC-SyAD-1003], [iTC\_CC\_ATP\_SwHA-0190]

[End]

[iTC\_CC\_ATP-SwRS-0139]

DriverInTrain，当前是否有司机在车内

If the active status is different between two ENDs of the train, ATP consider there is a driver in train.

def [DriverInTrain](#DriverInTrain)(k):

return ([DriverInCab\_1](#DriverInCab_1)(k) is not [DriverInCab\_2](#DriverInCab_2)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0348], [iTC\_CC\_ATP\_SwHA-0038]

[End]

[iTC\_CC\_ATP-SwRS-0127]

NonVitalSelectedFrontEnd，来自CCNV的车头选择信息

[NonVitalSelectedFrontEnd](#NonVitalSelectedFrontEnd) represents the train front choice from CCNV.

if ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k) == **True**)

[NonVitalSelectedFrontEnd](#NonVitalSelectedFrontEnd) = [NonVitalRequest](#NonVitalRequest).SelectedFrontEnd(k)

else:

[NonVitalSelectedFrontEnd](#NonVitalSelectedFrontEnd) = **UNKNOW**

#Category= Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1044]

[End]

[iTC\_CC\_ATP-SwRS-0138]

TrainFrontEnd，判断司机在**END\_1**还是**END\_2**还是由CCNV选择。

* If there is a driver in the train, the train front is the activated END. or else: the front determined by CCNV.
* Otherwise, the train front is the default one or the front one when train is moving.

def [TrainFrontEnd](#TrainFrontEnd)(k):

if (Initialization):

return **END\_2**

elif ([DriverInTrain](#DriverInTrain)(k)):

if ([DriverInCab\_1](#DriverInCab_1)(k)):

return **END\_1**

else:

return **END\_2**

elif ([NonVitalSelectedFrontEnd](#NonVitalSelectedFrontEnd)(k) is **END\_1**

or [NonVitalSelectedFrontEnd](#NonVitalSelectedFrontEnd)(k) is **END\_2**):

return [NonVitalSelectedFrontEnd](#NonVitalSelectedFrontEnd)(k)

elif ([WheelFilteredStopped](#WheelFilteredStopped)(k)):

return [TrainFrontEnd](#TrainFrontEnd)(k-1)

elif (not [End2RunningForward](#End2RunningForward)(k)):

return **END\_1**

else:

return **END\_2**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0203], [iTC\_CC-SyAD-0204], [iTC\_CC-SyAD-0345], [iTC\_CC-SyAD-1176], [iTC\_CC\_ATP\_SwHA-0191]

[End]

[iTC\_CC\_ATP-SwRS-0281]

TrainFrontOrientation，列车运营方向.

The train front orientation is the orientation of the active train END.

def [TrainFrontOrientation](#TrainFrontOrientation)(k):

if ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_2**):

return [TrainLocation](#TrainLocation).Ext2.Ort(k)

else:

return [TrainLocation](#TrainLocation).Ext1.Ort(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0177], [iTC\_CC-SyAD-0211]

[End]

[iTC\_CC\_ATP-SwRS-0249]

TrainFrontLocation，车头定位的更新:

ATP updates the train front location according to the active train END.

def [TrainFrontLocation](#TrainFrontLocation)(k):

if ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**):

[TrainFrontLocation](#TrainFrontLocation).Max = [TrainLocation](#TrainLocation).Ext1

[TrainFrontLocation](#TrainFrontLocation).Min = [TrainLocation](#TrainLocation).Int1

else:

[TrainFrontLocation](#TrainFrontLocation).Max = [TrainLocation](#TrainLocation).Ext2

[TrainFrontLocation](#TrainFrontLocation).Min = [TrainLocation](#TrainLocation).Int2

return [TrainFrontLocation](#TrainFrontLocation)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0211]

[End]

[iTC\_CC\_ATP-SwRS-0255]

TrainRearLocation，车尾定位的更新:

ATP updates the train rear locations according to the active train END.

def [TrainRearLocation](#TrainRearLocation)(k):

if ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**):

[TrainRearLocation](#TrainRearLocation).Max = [TrainLocation](#TrainLocation).Int2

[TrainRearLocation](#TrainRearLocation).Min = [TrainLocation](#TrainLocation).Ext2

else:

[TrainRearLocation](#TrainRearLocation).Max = [TrainLocation](#TrainLocation).Int1

[TrainRearLocation](#TrainRearLocation).Min = [TrainLocation](#TrainLocation).Ext1

return [TrainRearLocation](#TrainRearLocation)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0211]

[End]

#### Location path known status

[iTC\_CC\_ATP-SwRS-0757]

LocationPathKnown，判断列车在Block模式没有移动授权条件下是否经过道岔导致非确认定位。

In BM mode, if train localized but EOA is invalid, [LocationPathKnown](#LocationPathKnown) cannot be **TRUE** when train cross switch. After train crossed switch and relocalized by beacon successfully, [LocationPathKnown](#LocationPathKnown) can be set to **TRUE**.

def [LocationPathKnown](#LocationPathKnown)(k):

if (Initialization

or ([BlockModeUsed](#BlockModeUsed)(k)

and [TrainLocalized](#TrainLocalized)(k)

and not [BlockModeEOAvalid](#BlockModeEOAvalid)(k-1)

and not [TrainFilteredStopped](#TrainFilteredStopped)(k)

and [TrackMap.ExistSwitchBtwTwoLocs](#ExistSwitchBtwTwoLocs)([TrainRearLocation](#TrainRearLocation)(k).Min,

[TrainFrontLocation](#TrainFrontLocation)(k).Max))):

return **False**

elif ([TrainLocatedOnBeacon](#TrainLocatedOnBeacon)(k)

or [TrainRealignmentOnBeacon](#TrainRealignmentOnBeacon)(k)

or [TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP)(k)

or ([TrainPresumablyLocalized](#TrainPresumablyLocalized)(k) and not [TrainHasMoved](#TrainHasMoved)(k))):

return **True**

else:

return [LocationPathKnown](#LocationPathKnown)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1196], [iTC\_CC\_ATP\_SwHA-0248]

[End]

[iTC\_CC\_ATP-SwRS-0653]

TrainLocatedOnKnownPath，判断列车是否定位并已知[LocationPathKnown](#LocationPathKnown)。

def [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k):

return ([TrainLocalized](#TrainLocalized)(k)

and [LocationPathKnown](#LocationPathKnown)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1196]

[End]

**NOTES:**

BM下的EOA与[LocationPathKnown](#LocationPathKnown)和[TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)的关系如下：

* BM模式下，只要是之前BM的EOA无效，那么在岔区肯定不能使得[LocationPathKnown](#LocationPathKnown)为**True**；从而[TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)为**False**，BM的EOA也不可能有效；
* 当出了岔区，又读到信标进行重定位，或者通过记忆定位或远端定位初始化后，[LocationPathKnown](#LocationPathKnown)才能重新变为**True** ；
* 而如果之前BM的EOA是有效的，那么进入岔区时[LocationPathKnown](#LocationPathKnown)也不会变为无效。

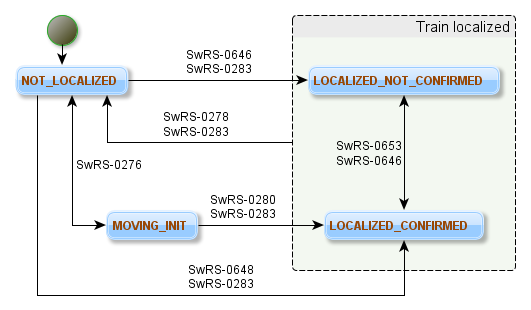


Figure 5‑15 Train localization state

列车的定位状态及其相关需求编号，如Figure 5‑15 Train localization state所示，有以下四种状态：

* **NOT\_LOCALIZED**，初始非定位状态。
* **MOVING\_INIT**，读到一个信标，进入定位初始化状态。
* **LOCALIZED\_NOT\_CONFIRMED**，已定位，但未确认状态。
* **LOCALIZED\_CONFIRMED**，已定位，且已确认定位。

[iTC\_CC\_ATP-SwRS-0275]

LocalizationState，列车的定位状态，用于用于维护诊断功能。

def [LocalizationState](#LocalizationState)(k):

if ([TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k)

and not [MemLocationNotConfirmed](#MemLocationNotConfirmed)(k)):

return **LOCALIZED\_CONFIRMED**

elif ([TrainLocalized](#TrainLocalized)(k)):

return **LOCALIZED\_NOT\_CONFIRMED**

elif ([MovingInitialByBeacon](#MovingInitialByBeacon)(k)):

return **MOVING\_INIT**

else:

return **NOT\_LOCALIZED**

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0408]

[End]

#### Location confirmation for ZC

[iTC\_CC\_ATP-SwRS-0654]

MemLocationNotConfirmed，是否通过本端或远端的记忆定位初始化列车定位，但还未通过重定位信标确认定位。

def [MemLocationNotConfirmed](#MemLocationNotConfirmed)(k):

if (Initialization

or not [TrainLocalized](#TrainLocalized)(k)

or [TrainRealignmentOnBeacon](#TrainRealignmentOnBeacon)(k)):

return **False**

elif ([TrainPresumablyLocalized](#TrainPresumablyLocalized)(k)):

return **True**

elif ([TrainLocatedOnOtherATP](#TrainLocatedOnOtherATP)(k)):

return [OtherATP](#OtherATP).LocatedWithMemLocation(k)

else:

return [MemLocationNotConfirmed](#MemLocationNotConfirmed)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0213], [iTC\_CC-SyAD-1208]

[End]

[iTC\_CC\_ATP-SwRS-0655]

LocationUncertaintyExceedTime，记录超过最大定位误差的时间

def [LocationUncertaintyExceedTime](#LocationUncertaintyExceedTime)(k):

if (Initialization):

return 0

elif ([TrainLocalized](#TrainLocalized)(k)

and (([MemLocationNotConfirmed](#MemLocationNotConfirmed)(k) or not [LocationPathKnown](#LocationPathKnown)(k))

and ([TrainLocation](#TrainLocation)(k).Uncertainty

> [ATPsetting](#ATPsetting).LocationMaxUncertaintyNotConfirmed))):

return ([ATPtime](#ATPtime)(k) + [ATPsetting](#ATPsetting).LocReportValidityTime)

else:

return [LocationUncertaintyExceedTime](#LocationUncertaintyExceedTime)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0213], [iTC\_CC-SyAD-1215]

[End]

计算在误差超过最大范围后，是否经过了[LocReport](#LocReport)有效期。即表明ZC那边已确保不会再用旧的定位信息

[iTC\_CC\_ATP-SwRS-0656]

LocationNotUncertaintyExceed，判断是否还处在最大定位误差的确认时间内

def [LocationNotUncertaintyExceed](#LocationNotUncertaintyExceed)(k):

if (Initialization):

return **False**

elif ([TrainLocalized](#TrainLocalized)(k)):

return [Message.IsMoreRecent](#IsMoreRecent)([ATPtime](#ATPtime)(k), [LocationUncertaintyExceedTime](#LocationUncertaintyExceedTime)(k))

else:

return [LocationNotUncertaintyExceed](#LocationNotUncertaintyExceed)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0213], [iTC\_CC-SyAD-1215]

[End]

[iTC\_CC\_ATP-SwRS-0411]

LocalizedAuthorizationForSweepping，发给ZC的是否定位信息。

ATP shall send the current localization status to the [ZC](#ZC).

def [LocalizedAuthorizationForSweepping](#LocalizedAuthorizationForSweepping)(k):

return ([TrainLocalized](#TrainLocalized)(k)

and [LocationNotUncertaintyExceed](#LocationNotUncertaintyExceed)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0213], [iTC\_CC\_ATP\_SwHA-0211]

[End]

[iTC\_CC\_ATP-SwRS-0412]

TrainConfirmedLocalized，发给ZC的是否确认定位信息。

­­ATP shall send the status of the localization status whether confirmed.

def [TrainConfirmedLocalized](#TrainConfirmedLocalized)(k):

return ([LocationPathKnown](#LocationPathKnown)(k)

and not [MemLocationNotConfirmed](#MemLocationNotConfirmed)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-1097]

[End]

#### Memorizing train location

[iTC\_CC\_ATP-SwRS-0657]

TrainIncludedInSleepingZone，列车停车后定位完全所在的Sleeping zone

def [TrainIncludedInSleepingZone](#TrainIncludedInSleepingZone)(k):

if (not [TrainFilteredStopped](#TrainFilteredStopped)(k)):

return **None**

else:

for SleepZone in ([TrackMap](#TrackMap).[AllSingsBtwTwoLocs](#AllSingsBtwTwoLocs)(**SGL\_SLEEPING\_ZONE**,

[TrackMap.BlockOrigin](#BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrainFrontLocation](#TrainFrontLocation)(k).Max)):

if ([TrackMap](#TrackMap).[LocationInZone](#LocationInZone)([TrainFrontLocation](#TrainFrontLocation)(k).Max,

SleepZone.Location,

SleepZone.Length,

SleepZone.Orientation)

and [TrackMap](#TrackMap).[LocationInZone](#LocationInZone)([TrainRearLocation](#TrainRearLocation)(k).Min,

SleepZone.Location,

SleepZone.Length,

SleepZone.Orientation)):

return SleepZone

else:

continue:

return **None**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1199]

[End]

**NOTES：**

为避免重复写入记忆定位导致存储空间寿命缩短，由设计保证ATP仅在完全进入休眠区并停车后写一次记忆定位（反之，不满足该条件时，也只清除一次记忆定位）。此外，在列车上电后还未移动时，无需重新写入记忆定位（因为此时用的还是原来记忆的定位）。

[iTC\_CC\_ATP-SwRS-0659]

WritingMemLocRequest，是否写入记忆定位。

Only when train has moved and filtered stopped in sleeping zone, can ATP writing memorized location information.

def [WritingMemLocRequest](#WritingMemLocRequest)(k):

return ([TrainHasMoved](#TrainHasMoved)(k)

and [TrainIncludedInSleepingZone](#TrainIncludedInSleepingZone)(k) is not **None**

and [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1199], [iTC\_CC\_ATP\_SwHA-0249]

[End]

[iTC\_CC\_ATP-SwRS-0660]

ClearingMemLocRequest，是否清除记忆定位

When train has moved and does not fulfill the condition of writing memory location, ATP shall clear memorized location information.

def [ClearingMemLocRequest](#ClearingMemLocRequest)(k):

return (not [WritingMemLocRequest](#WritingMemLocRequest)(k) and [TrainHasMoved](#TrainHasMoved)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1199], [iTC\_CC-SyAD-1204], [iTC\_CC\_ATP\_SwHA-0250]

[End]

[iTC\_CC\_ATP-SwRS-0661]

MemLocWritten，写入记忆定位的内容

def [MemLocWritten](#MemLocWritten)(k):

if ([WritingMemLocRequest](#WritingMemLocRequest)(k)):

[MemLocWritten](#MemLocWritten).MemLocVersion = **MEM\_LOCATION\_VERSION**

[MemLocWritten](#MemLocWritten).SleepAreaId = [TrainIncludedInSleepingZone](#TrainIncludedInSleepingZone)(k).Id

[MemLocWritten](#MemLocWritten).SleepAreaVersion = [TrainIncludedInSleepingZone](#TrainIncludedInSleepingZone)(k).Version

[MemLocWritten](#MemLocWritten).TrainType = [TrainType](#TrainType)(k)

[MemLocWritten](#MemLocWritten).TrainId = [SubSystemId](#SubSystemId)(k)

[MemLocWritten](#MemLocWritten).Ext2 = [TrainLocation](#TrainLocation).Ext2

[MemLocWritten](#MemLocWritten).Ext1 = [TrainLocation](#TrainLocation).Ext1

[MemLocWritten](#MemLocWritten).Uncertainty = [TrainLocation](#TrainLocation).Uncertainty

[MemLocWritten](#MemLocWritten).TrainLength = [ATPsetting](#ATPsetting).LocationTrainLength

elif ([ClearingMemLocRequest](#ClearingMemLocRequest)(k)):

[MemLocWritten](#MemLocWritten).MemLocVersion = **None**

[MemLocWritten](#MemLocWritten).SleepAreaId = **None**

[MemLocWritten](#MemLocWritten).SleepAreaVersion = **None**

[MemLocWritten](#MemLocWritten).TrainType = **None**

[MemLocWritten](#MemLocWritten).TrainId = **None**

[MemLocWritten](#MemLocWritten).Ext2 = **None**

[MemLocWritten](#MemLocWritten).Ext1 = **None**

[MemLocWritten](#MemLocWritten).Uncertainty = **None**

[MemLocWritten](#MemLocWritten).TrainLength = None

else:

[MemLocWritten](#MemLocWritten) = [MemLocWritten](#MemLocWritten)(k-1)

return [MemLocWritten](#MemLocWritten)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1201], [iTC\_CC\_ATP\_SwHA-0249], [iTC\_CC\_ATP\_SwHA-0250]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [LocalizationState](#LocalizationState) | √ | √ | ENUM\_LOCALIZATION\_STATE |
| [LocalizedAuthorizationForSweepping](#LocalizedAuthorizationForSweepping) | √ | √ | BOOLEAN |
| [LocationNotUncertaintyExceed](#LocationNotUncertaintyExceed) | √ | √ | BOOLEAN |
| [MemLocationNotConfirmed](#MemLocationNotConfirmed) | √ | √ | BOOLEAN |
| [TrainConfirmedLocalized](#TrainConfirmedLocalized) | √ | √ | BOOLEAN |
| [TrainFrontEnd](#TrainFrontEnd) | √ | √ | ENUM\_CAB\_ID |
| [TrainFrontLocation](#TrainFrontLocation) | √ | √ | ST\_LOCATION\_UNIT |
| [TrainFrontOrientation](#TrainFrontOrientation) | √ | √ | ENUM\_DOT |
| [TrainIncludedInSleepingZone](#TrainIncludedInSleepingZone) | √ | √ | BOOLEAN |
| [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath) | √ | √ | BOOLEAN |
| [TrainRearLocation](#TrainRearLocation) | √ | √ | ST\_LOCATION\_UNIT |

## F4-Monitor Train Energy

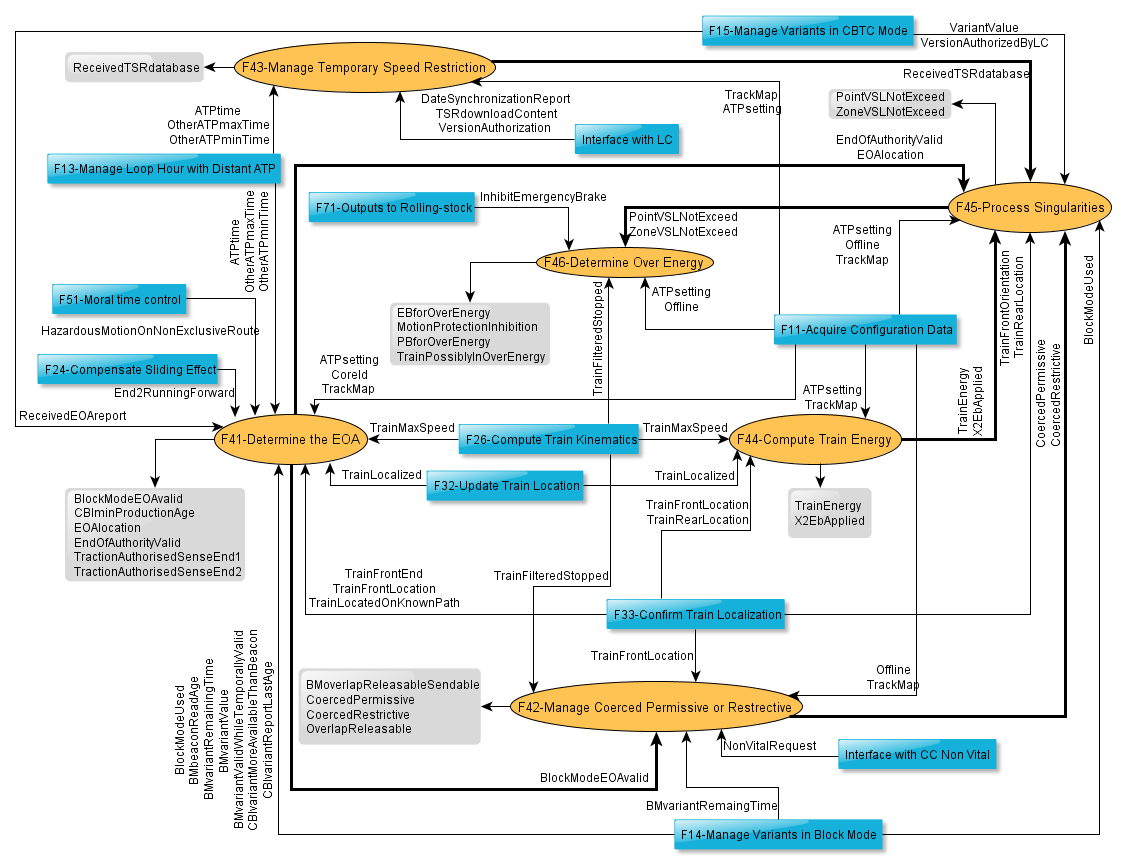


Figure 5‑16 SART modeling of function F4

## F41-Determine the EOA

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [ATPtime](#ATPtime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [BlockModeUsed](#BlockModeUsed) | Internal | F14-Manage Variants in Block Mode |
| [BMbeaconReadAge](#BMbeaconReadAge) | Internal | F14-Manage Variants in Block Mode |
| [BMvariantRemainingTime](#BMvariantRemainingTime) | Internal | F14-Manage Variants in Block Mode |
| [BMvariantValue](#BMvariantValue) | Internal | F14-Manage Variants in Block Mode |
| [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid) | Internal | F14-Manage Variants in Block Mode |
| [CBIvariantMoreAvailableThanBeacon](#CBIvariantMoreAvailableThanBeacon) | Internal | F14-Manage Variants in Block Mode |
| [CBIvariantReportLastAge](#CBIvariantReportLastAge) | Internal | F14-Manage Variants in Block Mode |
| [CoreId](#CoreId) | Internal | F11-Acquire Configuration Data |
| [End2RunningForward](#End2RunningForward) | Internal | F27-Compute Train Kinematics |
| [HazardousMotionOnNonExclusiveRoute](#HazardousMotionOnNonExclusiveRoute) | Internal | F51-Moral Time |
| [OtherATPmaxTime](#OtherATPmaxTime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [OtherATPminTime](#OtherATPminTime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [ReceivedEOAreport](#ReceivedEOAreport) | Internal | F15-Manage Variants in CBTC Mode |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFrontEnd](#TrainFrontEnd) | Internal | F33-Confirm Train Localization |
| [TrainFrontLocation](#TrainFrontLocation) | Internal | F33-Confirm Train Localization |
| [TrainLocalized](#TrainLocalized) | Internal | F32-Update Train Location |
| [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath) | Internal | F33-Confirm Train Localization |
| [TrainMaxSpeed](#TrainMaxSpeed) | Internal | F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [BlockModeEOAvalid](#BlockModeEOAvalid) | √ | √ | BOOLEAN |
| [CBIminProductionAgeSinceSSAcrossing](#CBIminProductionAgeSinceSSAcrossing) | √ | √ | NUMERIC\_32 |
| [ReceivedVariantsAfterEnteredBMinitialZone](#ReceivedVariantsAfterEnteredBMinitialZon) | √ | √ | BOOLEAN |
| [RestrictiveSignalOverrun](#RestrictiveSignalOverrun) | √ | √ | BOOLEAN |
| [StopAssuredPointCrossed](#StopAssuredPointCrossed) | √ | √ | BOOLEAN |
| [TrainEnteredInBMinitialZoneAge](#TrainEnteredInBMinitialZoneAge) | √ | √ | NUMERIC\_32 |
| [TrainInBMinitialZone](#TrainInBMinitialZone) | √ | √ | BOOLEAN |
| [TrainInSMIzone](#TrainInSMIzone) | √ | √ | BOOLEAN |

### Processing

ATP有CBTC和BLOCK两种运营模式，当采集到BM输入时，应用BM模式；否则，就应用CBTC模式。

There are two operational modes of ATP, BLOCK mode and CBTC. When ATP acquired BM input, it applies BM mode; otherwise, it applies CBTC mode.

ATP在BLOCK模式和CBTC模式下，对于EOA的处理方式不同。在CBTC模式，EOA来自ZC消息。而在BLOCK运营模式，ATP根据收到的BM变量状态，从列车所在定位位置，向下游进行搜索，当遇到第一个限制状态的变量或线路边界时，就将其作为本周期列车的EOA，该EOA的有效期也应当与该BM变量的有效期相同。

In the block model and CBTC model ATP have different processing mode for EOA. In the CBTC mode, EOA is coming from ZC message. In block mode, according to the BM variant status, ATP searches the target from downstream of the train front location. When it gets the first restricted variant or track border, ATP shall regard it at the EOA for this cycle, and the validity of EOA shall be the same as the BM variants.

#### Determing EOA in block mode

[iTC\_CC\_ATP-SwRS-0270]

RestrictiveSignalOverrun，BM模式下，本周期列车车头最大定位是否冒进限制状态的信号机。

* 当满足以下所有条件时，ATP认为列车冒进了限制状态的信号机，需设置RestrictiveSignalOverrun为**True**。
* 本周期列车已定位，即[TrainLocalized](#TrainLocalized)为**True**；
* 本周期使用BM变量；
* 上周期RestrictiveSignalOverrun为**False**；
* 本周期列车位移[MaximumTrainMotion](#MaximumTrainMotion)向激活的驾驶室方向运行；
* 本周期列车车头最大定位[TrainFrontLocation](#TrainFrontLocation)经过了一个信号机奇点；
* 该信号机为限制状态，或者建立了Overlap的状态。
* 否则，设置RestrictiveSignalOverrun为**False**。

[RestrictiveSignalOverrun](#RestrictiveSignalOverrun), ATP shall determine whether the location of maximum train head overruns a restricted signal in BLOCK mode.

* When all of the following conditions fulfilled, ATP considers the train has overrun a restricted signal in this cycle, and set [RestrictiveSignalOverrun](#RestrictiveSignalOverrun) as **True**.
* Train has localized;
* And the current type of EOA is **BLOCK\_MODE\_EOA**;
* And [RestrictiveSignalOverrun](#RestrictiveSignalOverrun) was **False** at the last cycle;
* And the moving direction in current cycle is toward on the train front end;
* And the maximum location of train front end passes the position of the signal in this cycle;
* And the status of the signal is restriction or overlap established.
* Otherwise, ATP set [RestrictiveSignalOverrun](#RestrictiveSignalOverrun) as **False**.

def [RestrictiveSignalOverrun](#RestrictiveSignalOverrun)(k):

sing = [TrackMap](#TrackMap).[ExistSingBtwTwoLocs](#ExistSingBtwTwoLocs)(**SGL\_SIGNAL**, [TrainFrontLocation](#TrainFrontLocation)(k-1).Max,

[TrainFrontLocation](#TrainFrontLocation)(k).Max)

return (sing is not **None**

and [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)(k)

and (([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_2** and [End2RunningForward](#End2RunningForward)(k))

or ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1** and [End1RunningForward](#End1RunningForward)(k)))

and not [BMvariantValue](#BMvariantValue)(sing.Variant, k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0290], [iTC\_CC\_ATP\_SwHA-0182]

[End]



Figure 5‑17 Train located in BM initialZone

BM模式的列车只有在BM初始化区域内才能获得移动授权，主要为防止列车下游到计轴边界范围内有未被ATP检测到的其他车辆。该区域定义为带BM初始化属性的信号机下游第一个Block边界，至边界上游[ATPsetting](#ATPsetting).BMinitAreaLength长度的区域，如Figure5-15所示。

注意，项目设计时，带有BM初始化属性的信号机奇点不能设置在运营方向的Block起始点处，而应设置在其上游Block的末端（实际上这两点是同一位置），就是说：

* 对于UP方向，BM初始化信号机不应设置在某Block的坐标0点处；
* 对于Down方向，BM初始化信号机不应设置在某Block的最大坐标（即坐标为Block长度）处。

[iTC\_CC\_ATP-SwRS-0662]

TrainInBMinitialZone，车头最小定位在在BM初始化区域内。

def [TrainInBMinitialZone](#TrainInBMinitialZone)(k):

NewBlock = [TrackMap.ExistSingularityInZone](#ExistSingularityInZone)(**SGL\_NEW\_BLOCK**, TrainFrontLocation(k).Min,

[ATPsetting](#ATPsetting).BMinitAreaLength)

Signal = ([TrackMap.ExistSingularityInReverseZone](#ExistSingularityInReverseZone)(**SGL\_SIGNAL**,

NewBlock.Location,

[ATPsetting](#ATPsetting).BMinitAreaLength))

if (Signal is not **None**

and Signal.BmInitialization):

return Signal

else:

return **None**

其中NewBlock.Location表示block的起始位置。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1284], [iTC\_CC-SyAD-0291]

[End]

[iTC\_CC\_ATP-SwRS-0663]

TrainEnteredInBMinitialZoneAge， 如果列车在BM初始化区域内，则记录已在该区域内运行的时间

def [TrainEnteredInBMinitialZoneAge](#TrainEnteredInBMinitialZoneAge)(k):

if ([TrainInBMinitialZone](#TrainInBMinitialZone)(k) is **None**):

[TrainEnteredInBMinitialZoneAge](#TrainEnteredInBMinitialZoneAge) = 0

else:

[TrainEnteredInBMinitialZoneAge](#TrainEnteredInBMinitialZoneAge) = [TrainEnteredInBMinitialZoneAge](#TrainEnteredInBMinitialZoneAge)(k-1) + 1

return [TrainEnteredInBMinitialZoneAge](#TrainEnteredInBMinitialZoneAge)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1284]

[End]

[iTC\_CC\_ATP-SwRS-0664]

StopAssuredPointCrossed，本周期是否通过了信号机前方的BMCP点

def [StopAssuredPointCrossed](#StopAssuredPointCrossed)(Cbi, k):

Bmcp = [TrackMap.ExistSingBtwTwoLocs](#ExistSingBtwTwoLocs)(**SGL\_BMCP**, [TrainFrontLocation](#TrainFrontLocation)(k-1).Max,

[TrainFrontLocation](#TrainFrontLocation)(k).Max)

return (Bmcp is not **None**

and cbi == Bmcp.CbiId)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1177]

[End]

[iTC\_CC\_ATP-SwRS-0665]

CBIminProductionAgeSinceSSAcrossing，记录从通过上个BMCP点开始到现在已经过了多长时间

def [CBIminProductionAgeSinceSSAcrossing](#CBIminProductionAgeSinceSSAcrossing)(Cbi, k):

if (Initialization

or [CBIminProductionAgeSinceSSAcrossing](#CBIminProductionAgeSinceSSAcrossing)(k-1)>= **REPORT\_AGE\_MAX**):

return **REPORT\_AGE\_MAX**

elif ([StopAssuredPointCrossed](#StopAssuredPointCrossed)(Cbi, k)):

return [ATPsetting](#ATPsetting).VariantsBMALSpresenceTimer

else:

return ([CBIminProductionAgeSinceSSAcrossing](#CBIminProductionAgeSinceSSAcrossing)(Cbi, k-1) + 1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1177]

[End]

[iTC\_CC\_ATP-SwRS-0666]

CBIminProductionAge，对于每个联锁，ATP维护最后收到其变量消息时联锁的最小时间，到现在经过的时间。

def [CBIminProductionAge](#CBIminProductionAge)(cbi, k):

return [min](#min)([CBIminProductionAgeSinceSSAcrossing](#CBIminProductionAgeSinceSSAcrossing)(Cbi, k),

[CBIvariantReportLastAge](#CBIvariantReportLastAge)(Cbi, k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1175]

[End]

[iTC\_CC\_ATP-SwRS-0667]

ReceivedVariantsAfterEnteredBMinitialZone，先进入BM初始化区，再收到无线或者信标的变量

def [ReceivedVariantsAfterEnteredBMinitialZone](#ReceivedVariantsAfterEnteredBMinitialZon)(k):

CbiId = [TrackMap.CbiId](#CbiId)([TrainInBMinitialZone](#TrainInBMinitialZone)(k).Block)

return ([TrainInBMinitialZone](#TrainInBMinitialZone)(k) is not **None**

and (([CBIvariantMoreAvailableThanBeacon](#CBIvariantMoreAvailableThanBeacon)(CbiId, k)

and (([CBIvariantReportLastAge](#CBIvariantReportLastAge)(CbiId, k)

+ [ATPsetting](#ATPsetting).VariantsBMproductionLatencyRadio)

< [TrainEnteredInBMinitialZoneAge](#TrainEnteredInBMinitialZoneAge)(k)))

or ([BMbeaconReadAge](#BMbeaconReadAge)(k) + [ATPsetting](#ATPsetting).VariantsBMproductionLatencyBeacon

< [TrainEnteredInBMinitialZoneAge](#TrainEnteredInBMinitialZoneAge)(k))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1284]

[End]

[iTC\_CC\_ATP-SwRS-0504]

BlockModeEOAvalid，BM下的移动授权是否可用

def [BlockModeEOAvalid](#BlockModeEOAvalid)(k):

if (Initialization

or not [BlockModeUsed](#BlockModeUsed)(k)

or [TrainFrontEnd](#TrainFrontEnd)(k) is not [TrainFrontEnd](#TrainFrontEnd)(k-1)

or not [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k)

or [HazardousMotionOnNonExclusiveRoute](#HazardousMotionOnNonExclusiveRoute)(k)

or [RestrictiveSignalOverrun](#RestrictiveSignalOverrun)(k)):

return **False**

elif (not [BlockModeEOAvalid](#BlockModeEOAvalid)(k-1)

and [TrainInBMinitialZone](#TrainInBMinitialZone)(k) is not **None**

and [BMvariantValue](#BMvariantValue)([TrainInBMinitialZone](#TrainInBMinitialZone).Variant(k), k)

and [ReceivedVariantsAfterEnteredBMinitialZone](#ReceivedVariantsAfterEnteredBMinitialZon)(k)):

return **True**

else:

return [BlockModeEOAvalid](#BlockModeEOAvalid)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0290], [iTC\_CC-SyAD-0293], [iTC\_CC-SyAD-1037], [iTC\_CC-SyAD-0291], [iTC\_CC-SyAD-0292], [iTC\_CC\_ATP\_SwHA-0047], [iTC\_CC\_ATP\_SwHA-0048], [iTC\_CC\_ATP\_SwHA-0050]

[End]

#### Determing EOA in CBTC mode

[iTC\_CC\_ATP-SwRS-0669]

TrainInSMIzone，判断当车头最大定位在SMI区域内，且车速小于SMI限速时，可使用ZC的EOA消息中的WithoutSpacingEoa进行监控。

def [TrainInSMIzone](#TrainInSMIzone)(k):

Smi = [TrackMap](#TrackMap).[ExistZoneLocationIncluded](#ExistZoneLocationIncluded)(**SGL\_SMI\_ZONE**, [TrainFrontEnd](#TrainFrontEnd)(k).Max)

return (Smi is not **None**

and [TrainMaxSpeed](#TrainMaxSpeed)(k) < Smi.SpeedLimit(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1281], [iTC\_CC-SyAD-1282], [iTC\_CC-SyAD-1283]

[End]

[iTC\_CC\_ATP-SwRS-0160]

CBTCmodeEOAvalid，CBTC模式下判断来自ZC的EOA是否有效。

* 如果在SMI区域内且车速小于SMI限速，则应当使用WithoutSpacingEOA；
* 否则，应当使用普通的EOA

def [CBTCmodeEOAvalid](#CBTCmodeEOAvalid)(k):

return (not [BlockModeUsed](#BlockModeUsed)(k)

and [**ReceivedEOAreport**](#ReceivedEOAreport).TrainFrontEnd == [TrainFrontEnd](#TrainFrontEnd)(k)

and (([TrainInSMIzone](#TrainInSMIzone)(k)

and ([Message.IsMoreRecent](#IsMoreRecent)([ReceivedEOAreport](#ReceivedEOAreport)(k).WithoutSpacing.ValidityTime,

[ATPtime](#ATPtime)(k)))

and ([ReceivedEOAreport](#ReceivedEOAreport)(k).WithoutSpacing.Location.Block != 0)

and ([**TrackMap**](#TrackMap).[DistanceBtwTwoLocs](#DistanceBtwTwoLocs)([**TrainFrontLocation**](#TrainFrontLocation)(k).Min,

[ReceivedEOAreport](#ReceivedEOAreport)(k).WithoutSpacing.Location,

[ATPsetting](#ATPsetting).EOAmaxDistance) is not **None**))

or (not [TrainInSMIzone](#TrainInSMIzone)(k)

and [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k)

and ([Message.IsMoreRecent](#IsMoreRecent)([ReceivedEOAreport](#ReceivedEOAreport)(k).Classic.ValidityTime,

[ATPtime](#ATPtime)(k)))

and ([ReceivedEOAreport](#ReceivedEOAreport)(k).Classic.Location.BlockId != 0)

and ([**TrackMap**](#TrackMap).[DistanceBtwTwoLocs](#DistanceBtwTwoLocs)([**TrainFrontLocation**](#TrainFrontLocation)(k).Min,

[ReceivedEOAreport](#ReceivedEOAreport)(k).Eoa.Location,

[ATPsetting](#ATPsetting).EOAmaxDistance) is not **None**))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0154], [iTC\_CC-SyAD-0157], [iTC\_CC-SyAD-0287], [iTC\_CC-SyAD-0288], [iTC\_CC-SyAD-0289], [iTC\_CC-SyAD-0293], [iTC\_CC-SyAD-0299], [iTC\_CC-SyAD-0842], [iTC\_CC-SyAD-1282], [iTC\_CC\_ATP\_SwHA-0028], [iTC\_CC\_ATP\_SwHA-0047], [iTC\_CC\_ATP\_SwHA-0052], [iTC\_CC\_ATP\_SwHA-0252]

[End]

**NOTES:**

对于普通EOA，ZC会检查发送的EOA坐标，确保其在Block长度范围内。而对于布置了SMI区的项目，为使得列车能尽量靠近轨道末端的车档停车，WithoutSpacing类型的EOA坐标可能为负值，或者大于所在Block长度，而其所在BlockID仍为轨道末端的Block。

[iTC\_CC\_ATP-SwRS-0670]

CBTCmodeEOAlocation，CBTC下的EOA位置。

def [CBTCmodeEOAlocation](#CBTCmodeEOAlocation)(k):

if ([CBTCmodeEOAvalid](#CBTCmodeEOAvalid)(k)):

if ([TrainInSMIzone](#TrainInSMIzone)(k)):

return [ReceivedEOAreport](#ReceivedEOAreport).WithoutSpacing.Location

else:

return [ReceivedEOAreport](#ReceivedEOAreport).Classic.Location

else:

return **None**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1282], [iTC\_CC-SyAD-1289], [iTC\_CC\_ATP\_SwHA-0252]

[End]

#### End of authority for train

[iTC\_CC\_ATP-SwRS-0671]

EndOfAuthorityValid，统一BM或CBTC下的EOA是否可用。

def [EndOfAuthorityValid](#EndOfAuthorityValid)(k):

if ([BlockModeUsed](#BlockModeUsed)(k)):

return [BlockModeEOAvalid](#BlockModeEOAvalid)(k)

else:

return [CBTCmodeEOAvalid](#CBTCmodeEOAvalid)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0289], [iTC\_CC-SyAD-0292], [iTC\_CC-SyAD-0293], [iTC\_CC-SyAD-0913]

[End]

#### Traction authorized direction

[iTC\_CC\_ATP-SwRS-0141]

TractionAuthorisedSenseEnd1，如果EOA有效且在**END\_1**方向，则ATP授权列车向**END\_1**方向运行。

If current EOA is valid and whose orientation is **END\_1**, ATP shall authorize the train can move toward **END\_1**.

def [TractionAuthorisedSenseEnd1](#TractionAuthorisedSenseEnd1)(k):

if ([EndOfAuthorityValid](#EndOfAuthorityValid)(k)

and [TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**):

[TractionAuthorisedSenseEnd1](#TractionAuthorisedSenseEnd1) = **True**

else:

[TractionAuthorisedSenseEnd1](#TractionAuthorisedSenseEnd1) = **False**

return [TractionAuthorisedSenseEnd1](#TractionAuthorisedSenseEnd1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0772]

[End]

[iTC\_CC\_ATP-SwRS-0142]

TractionAuthorisedSenseEnd2，如果EOA有效且在**END\_2**方向，则ATP授权向驾驶室2方向运行。

If current EOA is valid and whose orientation is **END\_2**, ATP shall authorize the train can move toward **END\_2**.

def [TractionAuthorisedSenseEnd2](#TractionAuthorisedSenseEnd2)(k):

if ([EndOfAuthorityValid](#EndOfAuthorityValid)(k)

and [TrainFrontEnd](#TrainFrontEnd)(k) is **END\_2**):

[TractionAuthorisedSenseEnd2](#TractionAuthorisedSenseEnd2) = **True**

else:

[TractionAuthorisedSenseEnd2](#TractionAuthorisedSenseEnd2) = **False**

return [TractionAuthorisedSenseEnd2](#TractionAuthorisedSenseEnd2)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0772]

[End]

### Outputs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Safety Critical | | Observable | | Logical Type |
| [BlockModeEOAvalid](#BlockModeEOAvalid) | | √ | | √ | BOOLEAN |
| [CBIminProductionAge](#CBIminProductionAge) | | √ | | √ | NUMERIC\_32 |
| [CBTCmodeEOAlocation](#CBTCmodeEOAlocation) | | √ | | √ | ST\_LOCATION\_UNIT |
| [CBTCmodeEOAvalid](#CBTCmodeEOAvalid) | | √ | | √ |  |
| [EndOfAuthorityValid](#EndOfAuthorityValid) | | √ | | √ |  |
| [TractionAuthorisedSenseEnd1](#TractionAuthorisedSenseEnd1) | | √ | | √ |  |
| [TractionAuthorisedSenseEnd2](#TractionAuthorisedSenseEnd2) | | √ | | √ |  |

## F42-Manage Coerced Permissive or Restrictive

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [BlockModeEOAvalid](#BlockModeEOAvalid) | Internal | F41-Determine the EOA |
| [BMvariantRemainingTime](#BMvariantRemainingTime) | Internal | F14-Manage Variants in Block Mode |
| [BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid) | Internal | F14-Manage Variants in Block Mode |
| [NonVitalRequest](#NonVitalRequest) | External | Interface with CC Non Vital |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainFrontLocation](#TrainFrontLocation) | Internal | F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [CoercedPermissive\_1](#CoercedPermissive_1) | √ | √ | BOOLEAN |
| [CoercedPermissive\_2](#CoercedPermissive_2) | √ | √ | BOOLEAN |
| [CoercedPermissive\_3](#CoercedPermissive_3) | √ | √ | BOOLEAN |
| [CoercedPermissive\_4](#CoercedPermissive_4) | √ | √ | BOOLEAN |
| [CrossedOverlapTimerInitialSignal](#CrossedOverlapTimerInitialSignal) | √ | √ | BOOLEAN |
| [NotCoercedRestrictive\_1](#NotCoercedRestrictive_1) | √ | √ | BOOLEAN |
| [NotCoercedRestrictive\_2](#NotCoercedRestrictive_2) | √ | √ | BOOLEAN |
| [NotCoercedRestrictive\_3](#NotCoercedRestrictive_3) | √ | √ | BOOLEAN |
| [NotCoercedRestrictive\_4](#NotCoercedRestrictive_4) | √ | √ | BOOLEAN |
| [OverlapTimer](#OverlapTimer) | √ | √ | NUMERIC\_32 |
| [OverlapTimerPermissive](#OverlapTimerPermissive) | √ | √ | BOOLEAN |

### Processing

#### Overlap releasable

1. Overlap状态管理

* 当选择BM模式，若[OverlapTimer](#OverlapTimer)大于0，则设置[OverlapTimerPermissive](#OverlapTimerPermissive)为True
* 若[OverlapTimerPermissive](#OverlapTimerPermissive)为**True**，则所有信号机的overlap强制建立

1. Overlap timer管理

* BM模式下，车头最大定位通过具有[OverlapTimer](#OverlapTimer)初始化属性的信号机，则设置[OverlapTimer](#OverlapTimer)为该信号机的变量有效期，并倒计时；当CC发送[OverlapReleasable](#OverlapReleasable)为允许状态时，并设置[OverlapTimer](#OverlapTimer)为0；当CC未授权在BM下运行时，也设置[OverlapTimer](#OverlapTimer)为0

[iTC\_CC\_ATP-SwRS-0600]

BMoverlapReleasableSendable，在BM下且未被ATC切除状态下，通过无线发给CI解锁信息。

def [BMoverlapReleasableSendable](#BMoverlapReleasableSendable)(k):

return [Offline.GetBMoverlapReleasableSendable](#GetBMoverlapReleasableSendable)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1313], [iTC\_CC\_ATP\_SwHA-0253]

[End]

[iTC\_CC\_ATP-SwRS-0673]

OverlapReleasable，可发送Overlap解锁信息

def [OverlapReleasable](#OverlapReleasable)(k):

return ([BMoverlapReleasableSendable](#BMoverlapReleasableSendable)(k)

and [TrainFilteredStopped](#TrainFilteredStopped)(k)

and [BlockModeEOAvalid](#BlockModeEOAvalid)(k)

and [NonVitalRequest](#NonVitalRequest).OverlapRelease(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1292], [iTC\_CC\_ATP\_SwHA-0254]

[End]

[iTC\_CC\_ATP-SwRS-0674]

CrossedOverlapTimerInitialSignal，即本周期通过一个overlap timer初始化信号机时，返回该信号机奇点

def [CrossedOverlapTimerInitialSignal](#CrossedOverlapTimerInitialSignal)(k):

Signal = [TrackMap.ExistSingBtwTwoLocs](#ExistSingBtwTwoLocs)(**SGL\_SIGNAL**, [TrainFrontLocation](#TrainFrontLocation)(k-1).Max,

[TrainFrontLocation](#TrainFrontLocation)(k).Max)

if (Signal is not **None**

and Signal.BmOverlapTimerInit):

return Signal

else:

return **None**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1189]

[End]

[iTC\_CC\_ATP-SwRS-0675]

OverlapTimer，当经过具有Overlap初始化属性的信号机时，将OverlapTimer设置为当时信号机的变量有效期。

def [OverlapTimer](#OverlapTimer)(k):

if (not [BlockModeEOAvalid](#BlockModeEOAvalid)(k)

or [OverlapReleasable](#OverlapReleasable)(k)):

return 0

elif ([BMvariantValidWhileTemporallyValid](#BMvariantValidWhileTemporallyValid)(k)

and [CrossedOverlapTimerInitialSignal](#CrossedOverlapTimerInitialSignal)(k) is not **None**):

return [[BMvariantRemainingTime](#BMvariantRemainingTime)](#BMvariantRemaingTime)([CrossedOverlapTimerInitialSignal](#CrossedOverlapTimerInitialSignal)(k).CBIvariant.Id, k)

else:

return [OverlapTimer](#OverlapTimer)(k-1) - 1

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1185], [iTC\_CC-SyAD-1189], [iTC\_CC-SyAD-1191], [iTC\_CC-SyAD-1192], [iTC\_CC\_ATP\_SwHA-0255]

[End]

[iTC\_CC\_ATP-SwRS-0676]

OverlapTimerPermissive，用于判断是否在BM下强制Overlap状态建立.

def [OverlapTimerPermissive](#OverlapTimerPermissive)(k):

return ([OverlapTimer](#OverlapTimer)(k) > 0)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1185], [iTC\_CC\_ATP\_SwHA-0255]

[End]

#### Coerced restrictive

在项目中可以在线路上的部分区域内配置“非强制限制”属性，来要求ATP在满足相应输入条件的情况下认为该区域是限制状态的。“非强制限制”状态的优先级要高于来自该区域变量的实际状态。非强制限制所支持的属性如Table 5‑7所示。

Table 5‑7 Configurable not coerced restrictive identification

|  |  |
| --- | --- |
| Not Coerced Restrictive | Functional Description |
| **VARIANTS\_RECEIVED\_FROM\_CBI\_ID** | 用于索引相关联锁的id，取值范围1~32 |
| **NOT\_COERCED\_RESTRICTIVE\_1** | 项目配置的非强制限制组合输入1 |
| **NOT\_COERCED\_RESTRICTIVE\_2** | 项目配置的非强制限制组合输入2 |
| **NOT\_COERCED\_RESTRICTIVE\_3** | 项目配置的非强制限制组合输入3 |
| **NOT\_COERCED\_RESTRICTIVE\_4** | 项目配置的非强制限制组合输入4 |

[iTC\_CC\_ATP-SwRS-0677]

NotCoercedRestrictive\_1，非强制限制1

def NotCoercedRestrictive\_1(k):

return [Offline](#Offline).[GetNotCoercedRestrictive\_1](#GetNotCoercedRestrictive)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1275], [iTC\_CC\_ATP\_SwHA-0256]

[End]

[iTC\_CC\_ATP-SwRS-0678]

NotCoercedRestrictive\_2，非强制限制2

def NotCoercedRestrictive\_2(k):

return [Offline](#Offline).[GetNotCoercedRestrictive\_2](#GetNotCoercedRestrictive)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1275], [iTC\_CC\_ATP\_SwHA-0256]

[End]

[iTC\_CC\_ATP-SwRS-0679]

NotCoercedRestrictive\_3，非强制限制3

def NotCoercedRestrictive\_3(k):

return [Offline](#Offline).[GetNotCoercedRestrictive\_3](#GetNotCoercedRestrictive)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1275], [iTC\_CC\_ATP\_SwHA-0256]

[End]

[iTC\_CC\_ATP-SwRS-0680]

NotCoercedRestrictive\_4，非强制限制4

def NotCoercedRestrictive\_4(k):

return [Offline](#Offline).[GetNotCoercedRestrictive\_4](#GetNotCoercedRestrictive)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1275], [iTC\_CC\_ATP\_SwHA-0256]

[End]

[iTC\_CC\_ATP-SwRS-0681]

CoercedRestrictive，等于相应的“非强制限制”取反。

def [CoercedRestrictive](#CoercedRestrictive)(ncr, k):

if (ncr is **NOT\_COERCED\_RESTRICTIVE\_1**):

[CoercedRestrictive](#CoercedRestrictive) = not [NotCoercedRestrictive\_1](#NotCoercedRestrictive_1)(k)

elif (ncr is **NOT\_COERCED\_RESTRICTIVE\_2**):

[CoercedRestrictive](#CoercedRestrictive) = not [NotCoercedRestrictive\_2](#NotCoercedRestrictive_2)(k)

elif (ncr is **NOT\_COERCED\_RESTRICTIVE\_3**):

[CoercedRestrictive](#CoercedRestrictive) = not [NotCoercedRestrictive\_3](#NotCoercedRestrictive_3)(k)

elif (ncr is **NOT\_COERCED\_RESTRICTIVE\_4**):

[CoercedRestrictive](#CoercedRestrictive) = not [NotCoercedRestrictive\_4](#NotCoercedRestrictive_4)(k)

elif (ncr is **VARIANTS\_RECEIVED\_FROM\_CBI\_ID**):

[CoercedRestrictive](#CoercedRestrictive) = not [CBIvariantLowValidity](#CBIvariantLowValidity)(**VARIANTS\_RECEIVED\_FROM\_CBI\_ID**, k)

else:

[CoercedRestrictive](#CoercedRestrictive) = **False**

return [CoercedRestrictive](#CoercedRestrictive)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1275], [iTC\_CC\_ATP\_SwHA-0256]

[End]

#### Coerced permissive

在项目上可以将线路上的部分区域（或点）设置“强制允许”属性，使得ATP判断在满足相应条件下，认为该区域（或点）为“强制允许”状态。“强制允许”的优先级要高于“非强制限制”和该区域（或点）对应的变量状态。可配置的“强制允许”属性如Table 5‑8所示。

Table 5‑8 Configurable coerced permissive identification

|  |  |
| --- | --- |
| Not Coerced Restrictive | Functional Description |
| **VARIANTS\_OVERLAP\_PERMISSIVE** | 用于在BM模式下强制建立信号机的Overlap |
| **COERCED\_PERMISSIVE\_1** | 项目配置的强制允许组合输入1 |
| **COERCED\_PERMISSIVE\_2** | 项目配置的强制允许组合输入2 |
| **COERCED\_PERMISSIVE\_3** | 项目配置的强制允许组合输入3 |
| **COERCED\_PERMISSIVE\_4** | 项目配置的强制允许组合输入4 |

[iTC\_CC\_ATP-SwRS-0682]

CoercedPermissive\_1，强制允许输入1

def [CoercedPermissive\_1](#CoercedPermissive_1) (k):

[CoercedPermissive\_1](#CoercedPermissive_1) = [Offline](#Offline).[GetCoercedPermissive\_1](#GetCoercedPermissive)(k)

return [CoercedPermissive\_1](#CoercedPermissive_1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1276], [iTC\_CC\_ATP\_SwHA-0257]

[End]

[iTC\_CC\_ATP-SwRS-0683]

CoercedPermissive\_2，强制允许输入2

def [CoercedPermissive\_2](#CoercedPermissive_2)(k):

[CoercedPermissive\_2](#CoercedPermissive_2) = [Offline](#Offline).[GetCoercedPermissive\_2](#GetCoercedPermissive)(k)

return [CoercedPermissive\_2](#CoercedPermissive_2)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1276], [iTC\_CC\_ATP\_SwHA-0257]

[End]

[iTC\_CC\_ATP-SwRS-0684]

CoercedPermissive\_3，强制允许输入3

def [CoercedPermissive\_3](#CoercedPermissive_3)(k):

[CoercedPermissive\_3](#CoercedPermissive_3) = [Offline](#Offline).[GetCoercedPermissive\_3](#GetCoercedPermissive)(k)

return [CoercedPermissive\_3](#CoercedPermissive_3)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1276], [iTC\_CC\_ATP\_SwHA-0257]

[End]

[iTC\_CC\_ATP-SwRS-0685]

CoercedPermissive\_4，强制允许输入4

def [CoercedPermissive\_4](#CoercedPermissive_4)(k):

[CoercedPermissive\_4](#CoercedPermissive_4) = [Offline](#Offline).[GetCoercedPermissive\_4](#GetCoercedPermissive)(k)

return [CoercedPermissive\_4](#CoercedPermissive_4)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1276], [iTC\_CC\_ATP\_SwHA-0257]

[End]

[iTC\_CC\_ATP-SwRS-0686]

CoercedPermissive，返回采集到的“强制允许”结果

def [CoercedPermissive](#CoercedPermissive)(cr, k):

if (cr is **COERCED\_PERMISSIVE\_1**):

[CoercedPermissive](#CoercedPermissive) = [CoercedPermissive\_1](#CoercedPermissive_1)(k)

elif (cr is **COERCED\_PERMISSIVE\_2**):

[CoercedPermissive](#CoercedPermissive) = [CoercedPermissive\_2](#CoercedPermissive_2)(k)

elif (cr is **COERCED\_PERMISSIVE\_3**):

[CoercedPermissive](#CoercedPermissive) = [CoercedPermissive\_3](#CoercedPermissive_3)(k)

elif (cr is **COERCED\_PERMISSIVE\_4**):

[CoercedPermissive](#CoercedPermissive) = [CoercedPermissive\_4](#CoercedPermissive_4)(k)

elif (cr is **VARIANTS\_OVERLAP\_PERMISSIVE**):

[CoercedPermissive](#CoercedPermissive) = [OverlapTimerPermissive](#OverlapTimerPermissive)(k)

else:

[CoercedPermissive](#CoercedPermissive) = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1187], [iTC\_CC-SyAD-1276], [iTC\_CC\_ATP\_SwHA-0257], [iTC\_CC\_ATP\_SwHA-0255]

[End]

### Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Safety Critical | | Observable | | | Logical Type |
| [BMoverlapReleasableSendable](#BMoverlapReleasableSendable) | | √ | | √ | BOOLEAN | |
| [CoercedPermissive](#CoercedPermissive) | | √ | | √ | BOOLEAN | |
| [CoercedRestrictive](#CoercedRestrictive) | | √ | | √ | BOOLEAN | |
| [OverlapReleasable](#OverlapReleasable) | | √ | | √ | BOOLEAN | |

## F43-Manage Temporary Speed Restriction

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [ATPtime](#ATPtime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [DateSynchronizationReport](#DateSynchronizationReport) | External | Interface with LC |
| [OtherATPmaxTime](#OtherATPmaxTime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [OtherATPminTime](#OtherATPminTime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TSRdownloadContent](#TSRdownloadContent) | External | Interface with LC |
| [VersionAuthorization](#VersionAuthorization) | External | Interface with LC |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [LastTSRreportAge](#LastTSRreportAge) | √ | √ | NUMERIC\_32 |
| [TSRreportAvailable](#TSRreportAvailable) | √ | √ | BOOLEAN |
| [TSRreportReceived](#TSRreportReceived) | √ | √ | BOOLEAN |

### Processing

#### Receiving TSR message

[iTC\_CC\_ATP-SwRS-0687]

TSRreportReceived，收到TSR消息

def [TSRreportReceived](#TSRreportReceived)(lcId, k):

return [Message.Received](#Received)([TSRdownloadContent](#TSRdownloadContent)(lcId), k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0282]

[End]

[iTC\_CC\_ATP-SwRS-0099]

TSRreportAvailable，TSR消息可用

def [TSRreportAvailable](#TSRreportAvailable)(lcId, k):

return Message.[Available](#Available)([TSRreportReceived](#TSRreportReceived)(lcId, k),

[TSRdownloadContent](#TSRdownloadContent).CcLoopHour,

[ATPsetting](#ATPsetting).TSRvalidityTime,

[LastTSRreportAge](#LastTSRreportAge)(lcId, k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0282], [iTC\_CC-SyAD-0391], [iTC\_CC\_ATP\_SwHA-0019]

[End]

[iTC\_CC\_ATP-SwRS-0688]

LastTSRreportAge，记录当前使用的TSR消息已经过了多长时间。

def [LastTSRreportAge](#LastTSRreportAge)(lcId, k):

return Message.[LastAge](#LastAge)([TSRreportAvailable](#TSRreportAvailable)(lcId, k),

[TSRdownloadContent](#TSRdownloadContent).CcLoopHour,

[LastTSRreportAge](#LastTSRreportAge)(lcId, k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0282]

[End]

**NOTES：**

由于TSR的解析和校核字计算需要一段时间，正常情况下，CCNV给LC消息发送的间隔应当大于TSR消息解析时间，确保在收到新消息时之前消息已解析完成。但如果在解析过程中又收到新的TSR消息时，应当遵循以下优先级处理：

* 应当继续解析当前的消息直至完成；
* 之后，选择与之前处理完成的消息所在不同的LC的消息进行解析；
* 对于每个LC里，只保留最新的一条消息，使用新消息覆盖旧的。

#### Parsing TSR

**NOTES**:

根据[REF5]，LC发送的TSR消息按照ST\_TSR\_DOWN\_CONTENT组织，但进行存储和SACEM校核时，需按照ST\_TSR\_BLOCK格式映射到线路地图上的每个BLOCK中。

* 对于每条TSR消息的中间BLOCK，其最小最大坐标分别为0和BLOCK长度值；
* 对于每条TSR消息的起始BLOCK：
* 如果该TSR区域是按照**UP**方向设置的，则转换后该BLOCK的最小坐标对应TSR的起始坐标；而最大坐标对应为该BLOCK长度值，或者TSR结束坐标（如果该TSR区域只包括这一个BLOCK）；
* 如果该TSR区域是按照**DOWN**方向设置的，则转换后该BLOCK的最小坐标对应为0，或TSR的结束坐标（如果该TSR区域只包括这一个BLOCK）；而最大坐标对应TSR的起始坐标。
* 对于每条TSR消息的结束BLOCK：
* 如果该TSR区域是按照**UP**方向设置的，则转换后该BLOCK的最小坐标对应为0，或TSR的起始坐标（如果该TSR区域只包括这一个BLOCK）；而最大坐标对应TSR的结束坐标。
* 如果该TSR区域是按照**DOWN**方向设置的，则转换后该BLOCK的最小坐标对应TSR的结束坐标；而最大坐标对应为该BLOCK长度值，或者TSR起始坐标（如果该TSR区域只包括这一个BLOCK）。

[iTC\_CC\_ATP-SwRS-0102]

ReceivedTSRdatabase，将LC发送的TSR消息报文映射到BLOCK数组中。对于线路上的每个BLOCK，判断其是否有对应的TSR，若有，则更新其首末点坐标和限速值，其中需将TSR消息中的坐标和速度单位转化为ATP软件使用的坐标和速度单位。

ATP shall map the TSR message received from LC to structure of block. It need to judge whether there is corresponding TSR for each BLOCK in the track map. If yes, ATP shall update the abscissa of the starting and ending points, as well as the restriction speed. During the process, it need to transfer the abscissa and speed unit of TSR message to the corresponding one used in ATP.

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ReceivedTSRdatabase | |  |  |
|  | ValidityTime | NUMERIC\_32 | Expiration time for the TSR message |
|  | Blocks[**MAX\_BLOCK\_NB**] | ST\_TSR\_BLOCK | TSR for each block |

def [ReceivedTSRdatabase](#ReceivedTSRdatabase)(lc, k):

if (Initialization

or (Message.[Exists](#Exists)([DateSynchronizationReport](#DateSynchronizationReport)(lc), k)

and Message.[Exists](#Exists)([VersionAuthorization](#VersionAuthorization)(lc), k)

and not Message.[Exists](#Exists)([TSRdownloadContent](#TSRdownloadContent)(lc), k))

or (not Message.[IsMoreRecent](#IsMoreRecent)([ReceivedTSRdatabase](#ReceivedTSRdatabase)(lc, k-1).ValidityTime, [ATPtime](#ATPtime)(k))

and not [TSRreportAvailable](#TSRreportAvailable)(k))):

[SetAllBlockAsDefaultTsr](#SetAllBlockAsDefaultTsr)

elif ([TSRreportAvailable](#TSRreportAvailable)(lc, k)):

NewValidity = 0

if ([Message.ReplyLocalCC](#ReplyLocalCC)([TSRdownloadContent](#TSRdownloadContent)(lc).CcLoopHour)):

NewValidity = ([TSRdownloadContent](#TSRdownloadContent)(lc).CcLoopHour + [ATPsetting](#ATPsetting).TSRvalidityTime)

else:

NewValidity = ([ATPtime](#ATPtime)(k) + [ATPsetting](#ATPsetting).TSRvalidityTime

- ([OtherATPmaxTime](#OtherATPmaxTime)(k) - [TSRdownloadContent](#TSRdownloadContent)(lc).CcLoopHour))

[ReceivedTSRdatabase](#ReceivedTSRdatabase).ValidityTime = NewValidity

for tsr in [range](#range)(0, [TSRdownloadContent](#TSRdownloadContent)(lc).NumberOfTsr):

[SetTsrInFirstBlock](#SetTsrInFirstBlock)

[SetTsrInLastBlock](#SetTsrInLastBlock)

for blk in [range](#range)([TSRdownloadContent](#TSRdownloadContent)(lc).Tsr[tsr].FirstBlockId + 1,

[TSRdownloadContent](#TSRdownloadContent)(lc).Tsr[tsr].LastBlockId):

[SetTsrInIntermediateBlock](#SetTsrInIntermediateBlock)

else:

[ReceivedTSRdatabase](#ReceivedTSRdatabase) = [ReceivedTSRdatabase](#ReceivedTSRdatabase)(lc, k-1)

return [ReceivedTSRdatabase](#ReceivedTSRdatabase)

其中，SetAllBlockAsDefaultTsr表示将线路**所有该LC管辖的Block**均设置为默认的TSR限速值[ATPsetting](#ATPsetting).TSRdefaultLimitSpeed；

SetTsrInFirstBlock表示TSR消息中首个Block的TSR设置；

SetTsrInIntermediateBlock表示TSR消息里中间Block的TSR设置。

SetTsrInLastBlock表示TSR消息中末尾Block的TSR设置。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0281], [iTC\_CC-SyAD-0283], [iTC\_CC-SyAD-0390], [iTC\_CC-SyAD-0392], [iTC\_CC-SyAD-0393], [iTC\_CC-SyAD-0914], [iTC\_CC-SyAD-1005], [iTC\_CC\_ATP\_SwHA-0022], [iTC\_CC\_ATP\_SwHA-0177], [iTC\_CC\_ATP\_SwHA-0178]

[End]

**NOTES**：

对于TSR的处理，仅支持一个BLOCK上至多有一个TSR的情况，其开始和结束点可以在该BLOCK上的任何位置。不支持一个BLOCK上有多个TSR。

For TSR processing, the iTC system supports only one TSR at one BLOCK at most. The beginning and the termination point of the TSR can be set any position in this block. However, it never sustains the situation that there are more than one TSR in one block.

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [ReceivedTSRdatabase](#ReceivedTSRdatabase) | √ | × | ST\_TSR\_BLOCK |

## F44-Compute Train Energy

本模块用于计算列车的能量。

This module calculates the train energy.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFrontLocation](#TrainFrontLocation) | Internal | F33-Confirm Train Localization |
| [TrainLocalized](#TrainLocalized) | Internal | F32-Update Train Location |
| [TrainMaxSpeed](#TrainMaxSpeed) | Internal | F27-Compute Train Kinematics |
| [TrainRearLocation](#TrainRearLocation) | Internal | F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [V1TractionCutoff](#V1TractionCutoff) | √ | √ | NUMERIC\_32 |
| [V2EbApplied](#V2EbApplied) | √ | √ | NUMERIC\_32 |
| [X1TractionCutoff](#X1TractionCutoff) | √ | √ | NUMERIC\_32 |

### Processing

文档[REF10]描述了列车能量的监控原理与实现方法，在EOA有效前提下，ATP根据过估的线路坡度和运动学属性，计算列车以当前速度和最大加速度条件下牵引切除紧急制动施加瞬间的速度和位置，作为列车的最大能量点。并以该点的能量，依次与车身范围及下游限制点的能量进行比较，判断列车是否超能。即如果列车超能，则ATP应当输出紧急制动，确保列车在最恶劣条件下不会超出线路限制速度以及冒进下游的EOA。

Document [REF10] describes the principle and implementation of the train energy monitoring. When the EOA is valid, ATP calculates the maximum energy position, where the traction has cut off and the brake begin to effect, based on the overestimated gradient and train kinematics. According to the principle of conservation of energy, ATP uses the train maximum energy to compare with the restricted energy of the vital zones train located or the downstream constraint points, to determine whether the current train energy exceeded the environment limits. If the over energy was detected, ATP shall request emergency braking to ensure that in the worst conditions, the train will not exceed the vital zone's speed limits or will not overrun the downstream EOA.

[iTC\_CC\_ATP-SwRS-0309]

X1TractionCutoff，V1TractionCutoff，在[EndOfAuthorityValid](#EndOfAuthorityValid)为**True**前提下，ATP根据列车当前最大速度[TrainMaxSpeed](#TrainMaxSpeed)，当前速度下的最大牵引力加速度（使用列车最小速度[TrainMinSpeed](#TrainMinSpeed)在配置数据中查找），车头最大定位所在Block的坡度最大加速度（来自配置数据），计算出经过时间后列车行驶的距离和所达到的速度。

If EOA is valid, ATP shall calculate the distance and the speed of the train moved after traction cutoff period (), according to the current train maximum speed, the maximum acceleration of traction and the maximum acceleration of the gradient.





其中，

* ，从ATP发出EB指令到列车牵引切除的时间[ATPsetting](#ATPsetting).EBtractionCutoffLatency
* 来自配置数据，是当前速度为时，列车牵引力所能提供的最大加速度[ATPsetting](#ATPsetting). TractionMaxAcc；
* 来自线路地图，是离线工具从当前所在block起始点上游[ATPsetting](#ATPsetting). LocationMaxUncertaintyConfirmed开始，记录的长度为（）的轨道区域内的坡度变化点中最大的坡度加速度。

Where,

* ，the latency from ATP trigger EB command to the traction cut off.
* , from [ATPsetting](#ATPsetting), is the maximum acceleration of traction at current speed.
* , from [TrackMap](#TrackMap), is the maximum acceleration of the gradient in the block start point train front end located plus to the distance ()

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0275], [iTC\_CC-SyAD-0306], [iTC\_CC-SyAD-0307], [iTC\_CC-SyAD-0315], [iTC\_CC\_ATP\_SwHA-0127], [iTC\_CC-SyAD-1004], [iTC\_CC-SyAD-0308]

[End]

[iTC\_CC\_ATP-SwRS-0310]

X2EbApplied，V2EbApplied，在[EndOfAuthorityValid](#EndOfAuthorityValid)为**True**前提下，根据车头最大定位所在Block的坡度最大加速度，计算出经过牵引切除(t1)和EB施加(t2)两部分时间后列车行驶的距离X2EbApplied和达到的速度V2EbApplied。

If EOA is valid, ATP shall calculate the distance and the speed of the train moved after the traction cutoff period plus emergency brake applied period (), according to the current train maximum speed, the maximum acceleration of traction and the maximum acceleration of the gradient.



Where,



* ，the [ATPsetting](#ATPsetting).EBtractionToBrakingLatency latency from RS cut off the traction to the emergency braking applied.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0275], [iTC\_CC-SyAD-0306], [iTC\_CC-SyAD-0315], [iTC\_CC\_ATP\_SwHA-0127], [iTC\_CC-SyAD-1004], [iTC\_CC-SyAD-0308]

[End]

[iTC\_CC\_ATP-SwRS-0312]

TrainEnergy，计算EB施加时刻的列车动能，作为能量监控使用的列车能量。

ATP shall calculate the train energy where EB indeed applied. The calculation shall consider the kinetic energy and the error of the potential energy.

[TrainEnergy](#TrainEnergy) = [V2EbApplied](#V2EbApplied) \* [V2EbApplied](#V2EbApplied)

+ [ATPsetting](#ATPsetting).MPauthAltitudeMaxErrorEnergy

The [ATPsetting](#ATPsetting).MPauthAltitudeMaxErrorEnergy means an algorithm error caused by offline tool to calculate the compensation gradients.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0275], [iTC\_CC-SyAD-0305], [iTC\_CC-SyAD-0306], [iTC\_CC-SyAD-0312], [iTC\_CC-SyAD-0313], [iTC\_CC\_ATP\_SwHA-0127]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [TrainEnergy](#TrainEnergy) | √ | √ | NUMERIC\_32 |
| [X2EbApplied](#X2EbApplied) | √ | √ | NUMERIC\_32 |

## F45-Process Singularities

对于已定位状态的列车，ATP需要根据车头或车尾所在的位置，处理线路上与列车位置有关奇点。例如判断列车是否在车站开门授权区域内，是否与PSD区域有交集，或者是否越过了限制状态的信号机等。

For the localized train, ATP shall consider the singularities within the range of train locations from the head END to the tail. For instance, ATP need to estimate whether the train is in the authorized area for opening the door in station, or whether the train location intersects with the PSD, or whether the train location exceeds the signal with restricted status.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [BlockModeUsed](#BlockModeUsed) | Internal | F14-Manage Variants in Block Mode |
| [CoercedPermissive](#CoercedPermissive) | Internal | F42-Manage Coerced Permissive or Restrictive |
| [CoercedRestrictive](#CoercedRestrictive) | Internal | F42-Manage Coerced Permissive or Restrictive |
| [EndOfAuthorityValid](#EndOfAuthorityValid) | Internal | F41-Determine the EOA |
| [[CBTCmodeEOAlocation](#CBTCmodeEOAlocation)](#EOAlocation) | Internal | F41-Determine the EOA |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [ReceivedTSRdatabase](#ReceivedTSRdatabase) | Internal | F43-Manage Temporary Speed Restriction |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainEnergy](#TrainEnergy) | Internal | F44-Compute Train Energy |
| [TrainFrontOrientation](#TrainFrontOrientation) | Internal | F33-Confirm Train Localization |
| [TrainRearLocation](#TrainRearLocation) | Internal | F33-Confirm Train Localization |
| [VariantValue](#VariantValue) | Internal | F15-Manage Variants in CBTC Mode |
| [VersionAuthorizedByLC](#VersionAuthorizedByLC) | Internal | F15-Manage Variants in CBTC Mode |
| [X2EbApplied](#X2EbApplied) | Internal | F44-Compute Train Energy |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [PointVSLNotExceedBSR](#PointVSLNotExceedBSR) | √ | √ | BOOLEAN |
| [PointVSLNotExceedCTE](#PointVSLNotExceedCTE) | √ | √ | BOOLEAN |
| [PointVSLNotExceedEOA](#PointVSLNotExceedEOA) | √ | √ | BOOLEAN |
| [PointVSLNotExceedOTE](#PointVSLNotExceedOTE) | √ | √ | BOOLEAN |
| [PointVSLNotExceedOverlap](#PointVSLNotExceedOverlap) | √ | √ | BOOLEAN |
| [PointVSLNotExceedPSD](#PointVSLNotExceedPSD) | √ | √ | BOOLEAN |
| [PointVSLNotExceedPZ](#PointVSLNotExceedPZ) | √ | √ | BOOLEAN |
| [PointVSLNotExceedSignal](#PointVSLNotExceedSignal) | √ | √ | BOOLEAN |
| [PointVSLNotExceedSwitch](#PointVSLNotExceedSwitch) | √ | √ | BOOLEAN |
| [PointVSLNotExceedTSR](#PointVSLNotExceedTSR) | √ | √ | BOOLEAN |
| [PointVSLNotExceedZC](#PointVSLNotExceedZC) | √ | √ | BOOLEAN |
| [TSRcontrolInhibition](#TSRcontrolInhibition) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedTrainSpeedLimit](#ZoneVSLNotExceedTrainSpeedLimit) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedBSR](#ZoneVSLNotExceedBSR) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedCTE](#ZoneVSLNotExceedCTE) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedEOA](#ZoneVSLNotExceedEOA) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedOTE](#ZoneVSLNotExceedOTE) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedOverlap](#ZoneVSLNotExceedOverlap) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedPSD](#ZoneVSLNotExceedPSD) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedPSR](#ZoneVSLNotExceedPSR) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedPZ](#ZoneVSLNotExceedPZ) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedSignal](#ZoneVSLNotExceedSignal) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedSwitch](#ZoneVSLNotExceedSwitch) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedTSR](#ZoneVSLNotExceedTSR) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceedZC](#ZoneVSLNotExceedZC) | √ | √ | BOOLEAN |

### Processing

对于与列车车身范围有交集的限制区域，例如PSR、TSR、限制状态的PSD或PZ等，ATP需保证列车的瞬时速度不超过该限制区域的限制值。此外，ATP还需保证列车车身范围不能与EOA、非受控道岔、轨道末端等有交集，否则将输出EB。

For the vital zone intersected with the train location, such as PSR, TSR, restricted PSD or Protection Zone etc., ATP shall inhibit the instantaneous velocity of the train will not greater than the limit. In addition, ATP shall request EB if the train location intersected with the EOA, uncontrolled point or track end.

ATP应当保证在最恶劣情况下，能使得列车停在下游的限制点前，即不会冒进EOA。同时，如果下游有降低的PSR或者TSR区域，ATP也应当能够保证列车以低于该限速的速度进入上述区域。为此，ATP需要计算下游各潜在限制点的动能和势能，确保当前列车能量低于限制点能量要求。

Even in the worst cases, ATP should ensure that the train could stop in the upstream of the constraint point, i.e. not exceeding EOA. Meanwhile, if there is PSR or TSR area in the downstream, ATP also should ensure trains enter the area below the speed restriction. Therefore, ATP needs to calculate kinetic energy and static energy of each potential limitation to ensure that the current train energy is lower than the limitation energy.

**NOTES：**

考虑到软件执行效率，ATP在比较车身范围限制区或下游限制点的能量时，当发现列车能量已经大于某限制区或限制点的能量时，可停止计算下游的奇点能量，直接返回超能结果。就是说，如果列车能量同时超过多个限制区或限制点时，ATP可能只会报出超过最近的一个限制区或限制点的能量。实际上，只要ATP检测出列车能量大于任意一个限制点或限制区，在非[MotionProtectionInhibition](#MotionProtectionInhibition)模式下，都将触发EB，致使列车停止。

#### Train speed limit

[iTC\_CC\_ATP-SwRS-0690]

ZoneVSLNotExceedTrainSpeedLimit，ATP应始终将项目配置的限速值为[ATPsetting](#ATPsetting).MPauthLimitSpeed作为安全速度限制区域。

* 限制区能量

def [ZoneVSLNotExceedTrainSpeedLimit](#ZoneVSLNotExceedTrainSpeedLimit)(k):

return ([**TrainEnergy**](#TrainEnergy)(k) < [pow](#pow)([**ATPsetting**](#ATPsetting).MPauthLimitSpeed))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0276], [iTC\_CC-SyAD-0300], [iTC\_CC\_ATP\_SwHA-0264]

[End]

#### PSR

PSR作为限制区域：列车车尾最小定位到EB实际施加位置的范围内的线路永久限速，需作为安全速度限制区域，其限速值为该PSR奇点中的属性值。

PSR作为限制点：限速减小的PSR起始点，其限速值为线路地图中该PSR奇点的属性值。

* 限制区能量
* 限制点能量



Figure 5‑18 PSR as vital speed limit zone

[iTC\_CC\_ATP-SwRS-0691]

ZoneVSLnotExceedPSR，PSR作为区域型限速的情形，ATP应将以下两种类型的PSR作F为限制区域进行监控：

* 该PSR是车尾最小定位上游的第一个PSR（即从该PSR所在位置到车尾最小定位之间没有其他PSR），如Figure 5‑18中的PSR2；
* 该PSR位于车尾最小定位下游到EB实际位置之间，如Figure 5‑18中的PSR2,PSR3和PSR4。

def [ZoneVSLnotExceedPSR](#ZoneVSLnotExceedPSR)(k):

for Psr in [TrackMap.AllSingsBtwTwoLocs](#AllSingsBtwTwoLocs)(**SGL\_PSR**,

[TrackMap.BlockOrigin](#TrackMap.BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k))):

if ([**TrainEnergy**](#TrainEnergy)(k) >= [pow](#pow)(Psr.SpeedLimit)

and (not [TrackMap.LocationBtwTwoLocs](#LocationBtwTwoLocs)(Psr.Location,

[TrackMap.BlockOrigin](#BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrainRearLocation](#TrainRearLocation)(k).Min)

or ([TrackMap.ExistSingBtwTwoLocs](#ExistSingBtwTwoLocs)(**SGL\_PSR**, Psr.Location,

[TrainRearLocation](#TrainRearLocation)(k).Min) is **None**))):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0284], [iTC\_CC-SyAD-0285], [iTC\_CC-SyAD-0300], [iTC\_CC\_ATP\_SwHA-0261]

[End]

[iTC\_CC\_ATP-SwRS-0692]

PointVSLnotExceedPSR，PSR作为点型限速的情形

def PointVSLnotExceedPSR(k):

for Psr in ([TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_PSR**,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

ATPsetting.EOAmaxDistance)):

if ([**TrainEnergy**](#TrainEnergy)(k) >= ([pow](#pow)(Psr.SpeedLimit)

+ ([[Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

Psr.Location)))):

return **False**

else:

continue

return **True**

其中，Energy.AccumulationPotentialEnergy表示根据限制点所在坡度或EB最小保障率累加计算目标位置的势能，EB最小保障率应根据所在位置的Grip值（Normal或Reduce）选取[ATPsetting](#ATPsetting).EBguaranteedAccNormalGrip或[ATPsetting](#ATPsetting).EBguaranteedAccReducedGrip。能量计算的原理和方法见[REF10]。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0284], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC-SyAD-0285], [iTC\_CC\_ATP\_SwHA-0261]

[End]

**NOTES:**

当车尾在一个较低的PSR（或TSR）中时，若当前车速小于该PSR限速，而计算出的V2速度大于该PSR限速，按照上述处理方式，也会导致EB，尽管当列车运行到X2位置时，列车也许已经离开了该PSR区域。

If the train tail intersected with a PSR (or TSR) area, and the speed of train is lower but the [V2EbApplied](#V2EbApplied) is higher than the limitation. In accordance with the above approach will result in EB, although when the train runs to the EB applied position, the train may have left the PSR area.

#### BSR

当ZC检测到计轴故障时，会自动激活该故障计轴所在的Block上的限速。ATP应检测车身范围及其下游Block的BSR变量，如果存在该变量且为限制状态，则认为该Block上的BSR激活，ATP应确保列车速度低于该限速。

[iTC\_CC\_ATP-SwRS-0693]

ZoneVSLnotExceedBSR，车身范围内有BSR的情形

def ZoneVSLnotExceedBSR(k):

for Block in ([TrackMap](#TrackMap).[AllSingsBtwTwoLocs](#AllSingsBtwTwoLocs)(**SGL\_NEW\_BLOCK**,

[TrackMap](#TrackMap).[BlockOrigin](#BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)))):

if (Block.Bsr is not **None**

and not [CoercedPermissive](#CoercedPermissive)(Block.[CoercedPermissive](#CoercedPermissive), k)

and not [VariantValue](#VariantValue)(Block.Bsr.Variant, k)

and [TrainEnergy](#TrainEnergy)(k) >= [pow](#pow)(Block.Bsr.Speed)):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-1267], [iTC\_CC-SyAD-1268], [iTC\_CC\_ATP\_SwHA-0261]

[End]

[iTC\_CC\_ATP-SwRS-0694]

PointVSLnotExceedBSR，列车下游有BSR的情形

def [PointVSLnotExceedBSR](#PointVSLnotExceedBSR)(k):

for Block in ([TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_NEW\_BLOCK**,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

ATPsetting.EOAmaxDistance)):

if (Block.Bsr is not **None**

and not [CoercedPermissive](#CoercedPermissive)(Block.Bsr.[CoercedPermissive](#CoercedPermissive), k)

and not [VariantValue](#VariantValue)(Block.Bsr.Variant, k)

and [TrainEnergy](#TrainEnergy)(k) >= ([pow](#pow)(Block.Bsr.Speed)

+ ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

Block.Bsr.Position)))):

return **False**

else:

continue

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC-SyAD-1267], [iTC\_CC-SyAD-1268], [iTC\_CC\_ATP\_SwHA-0261]

[End]

#### TSR

临时限速：

* 如果从列车车尾最小定位到EB实际施加位置的范围内存在临时限速，需作为安全速度限制区域，其限速值为从LC收到TSR消息中相应的速度值；
* 如果在EBA点下游存在限速减小的TSR，则ATP将其起始点作为安全限速点，其限速值为从LC收到TSR消息中相应的速度值。

[iTC\_CC\_ATP-SwRS-0069]

TSRcontrolInhibition，不处理TSR信息。其状态来自于项目可配置的列车输入采集。

According to the status of TSRcontrollinhibition, ATP can judge whether it is necessary to handle TSR information.

def [TSRcontrolInhibition](#TSRcontrolInhibition)(k):

return [Offline.GetTSRcontrolInhibition](#GetTSRcontrolInhibition)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0281], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-1003], [iTC\_CC-SyAD-1310], [iTC\_CC\_ATP\_SwHA-0205], [iTC\_CC\_ATP\_SwHA-0261]

[End]

[iTC\_CC\_ATP-SwRS-0695]

ZoneVSLnotExceedTSR，TSR作为区域型限速的情形。即对于从车尾所在Block起始点到EB施加位置内的所有Block，当满足以下条件时，认为列车超过了TSR限速：

* 未禁止处理TSR信息；
* 且该Block存在TSR；
* 且列车定位与该TSR区域有交集；
* 且计算的列车能量大于上述TSR的限制能量。

def ZoneVSLnotExceedTSR(k):

for Blk in ([TrackMap](#TrackMap).[AllSingsBtwTwoLocs](#AllSingsBtwTwoLocs)(**SGL\_NEW\_BLOCK**,

[TrackMap](#TrackMap).[BlockOrigin](#BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min)

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)))):

Tsr = [TSRonBlock](#TSRonBlock)(Blk, [TrackMap.OppositeOrientation](#OppositeOrientation)([TrainFrontOrientation](#TrainFrontOrientation)(k)), k)

if (not [TSRcontrolInhibition](#TSRcontrolInhibition)(k)

and Tsr is not **None**

and not [TrackMap](#TrackMap).[LocationBtwTwoLocs](#LocationBtwTwoLocs)(Tsr.Position,

[TrackMap](#TrackMap).[BlockOrigin](#BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrainRearLocation](#TrainRearLocation)(k).Min)

and [TrainEnergy](#TrainEnergy)(k) >= [pow](#pow)(Tsr.Value)):

return **False**

else:

continue

return **True**

其中TSRonBlock表示获取指定Block上TSR的值。

def [TSRonBlock](#TSRonBlock)(blockId, direction, k):

if (not [ReceivedTSRdatabase](#ReceivedTSRdatabase).[Blocks](#Blocks)[blockId].NotRestrictionApplication):

if (direction is UP):

return ([**ReceivedTSRdatabase**](#ReceivedTSRdatabase).[Blocks](#Blocks)[blockId].Position[0],

[ReceivedTSRdatabase](#ReceivedTSRdatabase).[Blocks](#Blocks)[blockId].Value)

else:

return ([ReceivedTSRdatabase](#ReceivedTSRdatabase).[Blocks](#Blocks)[blockId].Position[1],

[ReceivedTSRdatabase](#ReceivedTSRdatabase).[Blocks](#Blocks)[blockId].Value)

else:

return **None**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0285], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-1310], [iTC\_CC\_ATP\_SwHA-0261]

[End]

[iTC\_CC\_ATP-SwRS-0696]

PointVSLnotExceedTSR，TSR作为点型限速的情形

def [PointVSLnotExceedTSR](#PointVSLnotExceedTSR)(k):

for Blk in ([TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_NEW\_BLOCK**,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

ATPsetting.EOAmaxDistance)):

Tsr = [TSRonBlock](#TSRonBlock)(Blk, [TrainFrontOrientation](#TrainFrontOrientation)(k), k)

if (not [TSRcontrolInhibition](#TSRcontrolInhibition)(k)

and Tsr is not **None**

and [TrainEnergy](#TrainEnergy)(k) >= ([pow](#pow)(Tsr.Value)

+ ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

Tsr.Position)))):

return **False**

else:

continue

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0285], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC-SyAD-1310], [iTC\_CC\_ATP\_SwHA-0261]

[End]

#### Track ends

开放轨道终点：

* 当位于车头最大定位和EB实际施加位置之间时，ATP需将其作为安全速度限制区域处理，其限速值为0；
* 当位于EBA点下游时，作为安全限制点，限速为0。

[iTC\_CC\_ATP-SwRS-0697]

ZoneVSLnotExceedOTE，Open track end作为区域型限速的情形

def ZoneVSLnotExceedOTE(k):

if ([**TrackMap**.ExistSingularityInZone](#ExistSingularityInZone)(SGL\_OPEN\_TRACK\_END, [TrainFrontLocation](#TrainFrontLocation)(k).Max,

[X2EbApplied](#X2EbApplied)(k)) is not **None**)):

return **False**

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0304], [iTC\_CC\_ATP\_SwHA-0258]

[End]

[iTC\_CC\_ATP-SwRS-0698]

PointVSLnotExceedOTE，Open track end作为点型限速的情形

def PointVSLnotExceedOTE(k):

Ote = ([TrackMap](#TrackMap).[ExistSingularityInZone](#ExistSingularityInZone)(SGL\_OPEN\_TRACK\_END,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

[ATPsetting](#ATPsetting).EOAmaxDistance)

if (Ote is not **None**)

and [TrainEnergy](#TrainEnergy)(k) >= ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

Ote.Location))):

return **False**

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0304], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC\_ATP\_SwHA-0258]

[End]

封闭轨道终点：

* 当位于车头最大定位和EB实际施加位置之间时，ATP需将其作为安全速度限制区域处理，其限速值为该奇点的属性值.；
* 当位于EBA下游时，作为安全限速点，其限速为线路地图中描述的值。

[iTC\_CC\_ATP-SwRS-0699]

ZoneVSLnotExceedCTE，Close track end作为区域型限速的情形

def ZoneVSLnotExceedCTE(k):

cte = ([**TrackMap**](#TrackMap).[ExistSingularityInZone](#ExistSingularityInZone)(**SGL\_CLOSE\_TRACK\_END**, [TrainFrontLocation](#TrainFrontLocation)(k).Max,

[X2EbApplied](#X2EbApplied)(k)))

if (cte is not **None**

and [TrainEnergy](#TrainEnergy)(k) >= [pow](#pow)(cte.SpeedLimit)):

return **False**

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-1033], [iTC\_CC\_ATP\_SwHA-0258]

[End]

[iTC\_CC\_ATP-SwRS-0700]

PointVSLnotExceedCTE，Close track end作为点型限速的情形

def PointVSLnotExceedCTE(k):

cte = ([**TrackMap**](#TrackMap).[ExistSingularityInZone](#ExistSingularityInZone)(**SGL\_CLOSE\_TRACK\_END**,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

[ATPsetting](#ATPsetting).EOAmaxDistance))

if (cte is not **None**

and [TrainEnergy](#TrainEnergy)(k) >= ([[pow](#pow)](#pow)(cte.SpeedLimit)

+ ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

cte.Location)))):

return **False**

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC-SyAD-1033], [iTC\_CC\_ATP\_SwHA-0258]

[End]

#### Signals and overlap ends

根据项目配置的强制限制、强制允许以及信号机和Overlap自身变量状态的不同，ATP的能量监控点也不同，如所示。

Table 5‑9 Energy constraint point of signal and overlap end

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal | | Overlap | | Constraint Point |
| [CoercedRestrictive](#CoercedRestrictive) | [VariantValue](#VariantValue) | [CoercedPermissive](#CoercedPermissive) | [VariantValue](#VariantValue) |  |
| **True** | —— | **True** | —— | **SGL\_OVERLAP\_END** |
| **True** | —— | **False** | **True** | **SGL\_OVERLAP\_END** |
| **True** | —— | **False** | **False** | **SGL\_SIGNAL** |
| **False** | **True** | —— | —— | —— |
| **False** | **False** | **True** | —— | **SGL\_OVERLAP\_END** |
| **False** | **False** | **False** | **True** | **SGL\_OVERLAP\_END** |
| **False** | **False** | **False** | **False** | **SGL\_SIGNAL** |

信号机：

* 如果在车头最大定位和EB实际施加位置之间存在限制状态的信号机时，ATP需将其作为安全速度限制区域处理，其限速值为0；
* 如果在EBA点下游存在限制状态的信号机且未建立Overlap，其限速为0。

[iTC\_CC\_ATP-SwRS-0701]

ZoneVSLnotExceedSignal，信号机作为区域型限速的情形

def ZoneVSLnotExceedSignal(k):

for Sig in [TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_SIGNAL**, [TrainFrontLocation](#TrainFrontLocation)(k).Max, [X2EbApplied](#X2EbApplied)(k)):

if (([CoercedRestrictive](#CoercedRestrictive)(Sig.[NotCoercedRestrictive](#NotCoercedRestrictive), k)

or not [VariantValue](#VariantValue)(Sig.Variant, k))

and (not [CoercedPermissive](#CoercedPermissive)(Sig.[CoercedPermissive](#CoercedPermissive), k)

and not [VariantValue](#VariantValue)(Sig.OverlapVariant, k))):

return **False**

else:

continue

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-1286], [iTC\_CC\_ATP\_SwHA-0258]

[End]

[iTC\_CC\_ATP-SwRS-0702]

PointVSLnotExceedSignal，信号机作为点型限速的情形

def PointVSLnotExceedSignal(k):

for Sig in [TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_SIGNAL**,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

ATPsetting.EOAmaxDistance)):

if (([**CoercedRestrictive**](#CoercedRestrictive)(Sig.[NotCoercedRestrictive](#NotCoercedRestrictive), k)

or not [VariantValue](#VariantValue)(Sig.Variant, k))

and (not [CoercedPermissive](#CoercedPermissive)(Sig.[CoercedPermissive](#CoercedPermissive), k)

and not [VariantValue](#VariantValue)(Sig.OverlapVariant, k))):

and [TrainEnergy](#TrainEnergy)(k) >= ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

Sig.Location))):

return **False**

else:

continue

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC-SyAD-1286], [iTC\_CC\_ATP\_SwHA-0258]

[End]

Overlap终点：

* 如果在车头最大定位下游存在已建立Overlap的信号机，且相关的Overlap终点在列车车头最大定位下游和EB实际施加位置之间时，ATP将该Overlap终点作为安全速度限制区域处理，限速值为0；
* 如果在车头最大定位下游存在已建立Overlap的信号机，且相关的Overlap终点在EBA点下游，则该Overlap终点被视为安全速度限制点，其限速为0 。

[iTC\_CC\_ATP-SwRS-0703]

ZoneVSLnotExceedOverlap，Overlap作为区域型限速的情形

def ZoneVSLnotExceedOverlap(k):

for Overlap in [TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_OVERLAP\_END**, [TrainFrontLocation](#TrainFrontLocation)(k).Max,

[X2EbApplied](#X2EbApplied)(k)):

Signal = [TrackMap.ExistSingBtwTwoLocs](#ExistSingBtwTwoLocs)(SGL\_SIGNAL, [TrainFrontLocation](#TrainFrontLocation)(k).Max,

Overlap.Location)

if (Signal is not **None**

and ([**CoercedRestrictive**](#CoercedRestrictive)(Signal.[NotCoercedRestrictive](#NotCoercedRestrictive), k)

or not [VariantValue](#VariantValue)(Signal.Variant, k))

and ([CoercedPermissive](#CoercedPermissive)(Signal.[CoercedPermissive](#CoercedPermissive), k)

or [VariantValue](#VariantValue)(Overlap.Variant, k))):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-1187], [iTC\_CC\_ATP\_SwHA-0258]

[End]

[iTC\_CC\_ATP-SwRS-0704]

PointVSLnotExceedOverlap，Overlap作为点型限速的情形

def PointVSLnotExceedOverlap(k):

for Overlap in ([TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_OVERLAP\_END**,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

ATPsetting.EOAmaxDistance)):

Signal = ([TrackMap.ExistSingBtwTwoLocs](#ExistSingBtwTwoLocs)(SGL\_SIGNAL,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

Overlap.Location))

if (Signal is not **None**

and ([**CoercedRestrictive**](#CoercedRestrictive)(Signal.[NotCoercedRestrictive](#NotCoercedRestrictive), k)

or not [VariantValue](#VariantValue)(Signal.Variant, k))

and ([**CoercedPermissive**](#CoercedPermissive)(Signal.[CoercedPermissive](#CoercedPermissive), k)

or [VariantValue](#VariantValue)(Overlap.Variant, k))

and [TrainEnergy](#TrainEnergy)(k) >= ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

Overlap.Location))):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC-SyAD-1187], [iTC\_CC\_ATP\_SwHA-0258]

[End]

#### Uncontrolled points

道岔：

* 如果在车头最大定位和EB实际施加位置之间存在非受控状态的道岔（无论汇聚节点还是发散节点），ATP需将其作为安全速度限制区域处理，其限速值为0.
* 如果在EBA点下游存在非受控状态的道岔，ATP需将其作为安全速度限制点，其限速值为0；

[iTC\_CC\_ATP-SwRS-0705]

ZoneVSLnotExceedSwitch，非受控道岔作为区域型限速的情形

def ZoneVSLnotExceedSwitch(k):

for Switch in [TrackMap.AllSwitchesInZone](#AllSwitchesInZone)([TrainFrontLocation](#TrainFrontLocation)(k).Max, [X2EbApplied](#X2EbApplied)(k)):

if ([VariantValue](#VariantValue)(Switch.Variant1, k) == [VariantValue](#VariantValue)(Switch.Variant2, k)):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0868], [iTC\_CC\_ATP\_SwHA-0258]

[End]

[iTC\_CC\_ATP-SwRS-0706]

PointVSLnotExceedSwitch，非受控道岔作为点型限速的情形

def PointVSLnotExceedSwitch(k):

for Switch in ([TrackMap.AllSwitchesInZone](#AllSwitchesInZone)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

ATPsetting.EOAmaxDistance)):

if ([VariantValue](#VariantValue)(Switch.Variant1, k) == [VariantValue](#VariantValue)(Switch.Variant2, k)

and [TrainEnergy](#TrainEnergy)(k) >= ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

Switch.Location))):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC-SyAD-0868], [iTC\_CC\_ATP\_SwHA-0258]

[End]

#### Protection zones

保护区：

* 如果从列车车尾最小定位到EB实际施加位置的范围内存在限制状态的保护区，ATP需将其作为安全速度限制区域处理，其限速值为0；
* 如果在EBA下游存在限制状态的保护区起始点，其限速为0。

[iTC\_CC\_ATP-SwRS-0707]

ZoneVSLnotExceedPZ，PZ作为区域型限速。ATP应监控与列车定位有以下两种关系的限制状态保护区：

* 该保护区的起始点在车尾最小定位到紧急制动施加位置之间；
* 或，该保护区起始点在车尾最小定位上游，但车尾最小定位在该保护区范围内。

def ZoneVSLnotExceedPZ(k):

for Pz in ([TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_PROTECTION\_ZONE**,

[TrackMap.BlockOrigin](#TrackMap.BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)))):

if ((not [TrackMap.LocationBtwTwoLocs](#LocationBtwTwoLocs)(Pz.Location,

[TrackMap.BlockOrigin](#BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrainRearLocation](#TrainRearLocation)(k).Min)

or [TrackMap.LocationInZone](#LocationInZone)([TrainRearLocation](#TrainRearLocation)(k).Min, Pz.Location, Pz.Length))

and not [CoercedPermissive](#CoercedPermissive)(Pz.[CoercedPermissive](#CoercedPermissive), k)

and ([CoercedRestrictive](#CoercedRestrictive)(Pz.[NotCoercedRestrictive](#NotCoercedRestrictive), k)

or not [VariantValue](#VariantValue)(Pz.Variant, k))):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0225], [iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0796], [iTC\_CC-SyAD-1273], [iTC\_CC-SyAD-1274], [iTC\_CC-SyAD-1277], [iTC\_CC\_ATP\_SwHA-0258]

[End]

[iTC\_CC\_ATP-SwRS-0708]

PointVSLnotExceedPZ，PZ作为点型限速的情形

def PointVSLnotExceedPZ(k):

for Pz in ([TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_PROTECTION\_ZONE**,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

ATPsetting.EOAmaxDistance)):

if (not [CoercedPermissive](#CoercedPermissive)(Pz.[CoercedPermissive](#CoercedPermissive), k)

and ([CoercedRestrictive](#CoercedRestrictive)(Pz.[NotCoercedRestrictive](#NotCoercedRestrictive), k)

or not [VariantValue](#VariantValue)(Pz.Variant, k))

and [TrainEnergy](#TrainEnergy)(k) >= ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

Pz.Location))):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0225], [iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0295], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC-SyAD-0796], [iTC\_CC-SyAD-1273], [iTC\_CC-SyAD-1274], [iTC\_CC-SyAD-1277], [iTC\_CC\_ATP\_SwHA-0258]

[End]

#### PSD zones

屏蔽门区域：

* 如果从列车车尾最小定位到EB实际施加位置的范围内存在限制状态的PSD，ATP需将其作为安全速度限制区域处理，其限速值为0；
* 如果EBA下游存在限制状态的屏蔽门区起始点，其限速为0。

[iTC\_CC\_ATP-SwRS-0709]

ZoneVSLnotExceedPSD，PSD作为区域型限速。ATP应监控与列车定位有以下两种关系的限制状态屏蔽门区域：

* 该屏蔽门区的起始点在车尾最小定位到紧急制动施加位置之间；
* 或，该屏蔽门区起始点在车尾最小定位上游，但车尾最小定位在该屏蔽门区范围之内。

def ZoneVSLnotExceedPSD(k):

for Psd in ([TrackMap](#TrackMap).[AllSingsBtwTwoLocs](#AllSingsBtwTwoLocs)(**SGL\_PSD\_ZONE**,

[TrackMap.BlockOrigin](#TrackMap.BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)))):

if ((not [TrackMap.LocationBtwTwoLocs](#LocationBtwTwoLocs)(Psd.Location,

[TrackMap.BlockOrigin](#BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrainRearLocation](#TrainRearLocation)(k).Min)

or [TrackMap.LocationInZone](#LocationInZone)([TrainRearLocation](#TrainRearLocation)(k).Min, Psd.Location, Psd.Length))

and not [CoercedPermissive](#CoercedPermissive)(Psd.[CoercedPermissive](#CoercedPermissive), k)

and ([CoercedRestrictive](#CoercedRestrictive)(Psd.[NotCoercedRestrictive](#NotCoercedRestrictive), k)

or not [VariantValue](#VariantValue)(Psd.Variant, k))):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-1398], [iTC\_CC-SyAD-1399], [iTC\_CC\_ATP\_SwHA-0258]

[End]

[iTC\_CC\_ATP-SwRS-0710]

PointVSLnotExceedPSD，PSD作为点型限速的情形

def PointVSLnotExceedPSD(k):

for Psd in ([TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_PSD\_ZONE**,

[[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k) )](#EBAlocation),

ATPsetting.EOAmaxDistance)):

if (not [CoercedPermissive](#CoercedPermissive)(Psd.[CoercedPermissive](#CoercedPermissive), k)

and ([CoercedRestrictive](#CoercedRestrictive)(Psd.[NotCoercedRestrictive](#NotCoercedRestrictive), k)

or not [VariantValue](#VariantValue)(Psd.Variant, k))

and [TrainEnergy](#TrainEnergy)(k) >= ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

Psd.Location))):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0301], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC-SyAD-1398], [iTC\_CC-SyAD-1399], [iTC\_CC\_ATP\_SwHA-0258]

[End]

#### ZC zones without LC authorization

CBTC模式下非LC授权的ZC边界：

* CBTC模式下，如果从列车车尾最小定位到EB实际施加位置的范围内存在非LC授权的ZC区域，ATP需将其作为安全速度限制区域处理，其限速值为0；
* CBTC模式下，如果非授权的ZC区域边界在EBA点下游，则ATP将其作为安全限速点，限制点为0。

[iTC\_CC\_ATP-SwRS-0711]

ZoneVSLnotExceedZC，非授权ZC作为区域型限速的情形

def ZoneVSLnotExceedZC(k):

for block in ([TrackMap](#TrackMap).[AllSingsBtwTwoLocs](#AllSingsBtwTwoLocs)(**SGL\_NEW\_BLOCK**, [TrainRearLocation](#TrainRearLocation)(k).Min,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)))):

if (not [BlockModeUsed](#BlockModeUsed)(k)

and not [VersionAuthorizedByLC](#VersionAuthorizedByLC)([TrackMap](#TrackMap).[ZCId](#ZCId)(block.Id), k)):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0300], [iTC\_CC\_ATP\_SwHA-0258]

[End]

[iTC\_CC\_ATP-SwRS-0712]

PointVSLnotExceedZC，非授权ZC边界作为点型限速的情形

def PointVSLnotExceedZC(k):

for NewBlock in ([TrackMap.AllSingsInZone](#AllSingsInZone)(**SGL\_NEW\_BLOCK**,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k)),

ATPsetting.EOAmaxDistance)):

if (not [BlockModeUsed](#BlockModeUsed)(k)

and not [VersionAuthorizedByLC](#VersionAuthorizedByLC)([TrackMap](#TrackMap).[ZCId](#ZCId)(NewBlock.Id), k)

and [TrainEnergy](#TrainEnergy)(k) >= ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

NewBlock.Location))):

return **False**

else:

continue

return **True**

其中NewBlock.Location表示该block的起始位置。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC\_ATP\_SwHA-0258]

[End]

#### EOA

CBTC模式下，授权行驶终点应作为ATP的限制点：

* 如果从列车车头最小定位到EB实际施加位置的范围内存在来自ZC的EOA点，ATP需将其作为安全速度限制区域处理，其限速值为0；
* 如果来自ZC的EOA位于EBA点下游，则ATP将其作为安全限速点，限速为0.

[iTC\_CC\_ATP-SwRS-0713]

ZoneVSLnotExceedEOA，CBTC下EOA作为区域型限速的情形

def ZoneVSLnotExceedEOA(k):

if ([[CBTCmodeEOAvalid](#CBTCmodeEOAvalid)](#EndOfAuthorityValid)(k)

and ([TrackMap](#TrackMap).[LocationBtwTwoLocs](#LocationBtwTwoLocs)([[CBTCmodeEOAlocation](#CBTCmodeEOAlocation)](#EOAlocation)(k),

[TrainFrontLocation](#TrainFrontLocation)(k).Min,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k))))):

return **False**

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0289], [iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0300], [iTC\_CC\_ATP\_SwHA-0258]

[End]

[iTC\_CC\_ATP-SwRS-0714]

PointVSLnotExceedEOA，CBTC下EOA作为点型限速的情形

def PointVSLnotExceedEOA(k):

if ([[CBTCmodeEOAvalid](#CBTCmodeEOAvalid)](#EndOfAuthorityValid)(k)

and not ([TrackMap](#TrackMap).[LocationBtwTwoLocs](#LocationBtwTwoLocs)([[CBTCmodeEOAlocation](#CBTCmodeEOAlocation)](#EOAlocation)(k),

[TrainFrontLocation](#TrainFrontLocation)(k).Min,

[TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max, [X2EbApplied](#X2EbApplied)(k))))

and [TrainEnergy](#TrainEnergy)(k) >= ([Energy.AccumulationPotentialEnergy](#AccumulationPotentialEnergy)

([TrackMap.CalculateZoneBorder](#CalculateZoneBorder)([TrainFrontEnd](#TrainFrontEnd)(k).Max,

[X2EbApplied](#X2EbApplied)(k)),

[[CBTCmodeEOAlocation](#CBTCmodeEOAlocation)](#EOAlocation)(k)))):

return **False**

else:

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0289], [iTC\_CC-SyAD-0294], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0311], [iTC\_CC-SyAD-0314], [iTC\_CC\_ATP\_SwHA-0258]

[End]

#### Monitor train energy against restrictive zones or points

[iTC\_CC\_ATP-SwRS-0318]

ZoneVSLNotExceed，判断有无限制区域超能。

ATP shall determine whether train exceeds the vital speed limitation of the restrictive zone, by comparing the energy between the energy of the train and the energy of the zone.

def [ZoneVSLNotExceed](#ZoneVSLNotExceed)(k):

return ([EndOfAuthorityValid](#EndOfAuthorityValid)(k)

and [ZoneVSLNotExceedTrainSpeedLimit](#ZoneVSLNotExceedTrainSpeedLimit)(k)

and [ZoneVSLnotExceedPSR](#ZoneVSLnotExceedPSR)(k)

and [ZoneVSLnotExceedBSR](#ZoneVSLnotExceedBSR)(k)

and [ZoneVSLnotExceedTSR](#ZoneVSLnotExceedTSR)(k)

and [ZoneVSLnotExceedOTE](#ZoneVSLnotExceedOTE)(k)

and [ZoneVSLnotExceedCTE](#ZoneVSLnotExceedCTE)(k)

and [ZoneVSLnotExceedSignal](#ZoneVSLnotExceedSignal)(k)

and [ZoneVSLnotExceedOverlap](#ZoneVSLnotExceedOverlap)(k)

and [ZoneVSLnotExceedSwitch](#ZoneVSLnotExceedSwitch)(k)

and [ZoneVSLnotExceedPZ](#ZoneVSLnotExceedPZ)(k)

and [ZoneVSLnotExceedPSD](#ZoneVSLnotExceedPSD)(k)

and [ZoneVSLnotExceedZC](#ZoneVSLnotExceedZC)(k)

and [ZoneVSLnotExceedEOA](#ZoneVSLnotExceedEOA)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0227], [iTC\_CC-SyAD-0276], [iTC\_CC-SyAD-0293], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0316], [iTC\_CC-SyAD-0137], [iTC\_CC-SyAD-1270], [iTC\_CC-SyAD-0292], [iTC\_CC\_ATP\_SwHA-0259], [iTC\_CC\_ATP\_SwHA-0262]

[End]

[iTC\_CC\_ATP-SwRS-0324]

PointVSLNotExceed，判断有无限制点超能。

ATP shall determine whether train exceeds the vital speed limitation of the restrictive point, by comparing the energy between the energy of the train and the kinetic added potential energy of the point.

def [PointVSLNotExceed](#PointVSLNotExceed)(k):

return ([EndOfAuthorityValid](#EndOfAuthorityValid)(k)

and [PointVSLnotExceedPSR](#PointVSLnotExceedPSR)(k)

and [PointVSLnotExceedBSR](#PointVSLnotExceedBSR)(k)

and [PointVSLnotExceedTSR](#PointVSLnotExceedTSR)(k)

and [PointVSLnotExceedOTE](#PointVSLnotExceedOTE)(k)

and [PointVSLnotExceedCTE](#PointVSLnotExceedCTE)(k)

and [PointVSLnotExceedSignal](#PointVSLnotExceedSignal)(k)

and [PointVSLnotExceedOverlap](#PointVSLnotExceedOverlap)(k)

and [PointVSLnotExceedSwitch](#PointVSLnotExceedSwitch)(k)

and [PointVSLnotExceedPZ](#PointVSLnotExceedPZ)(k)

and [PointVSLnotExceedPSD](#PointVSLnotExceedPSD)(k)

and [PointVSLnotExceedZC](#PointVSLnotExceedZC)(k)

and [PointVSLnotExceedEOA](#PointVSLnotExceedEOA)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0293], [iTC\_CC-SyAD-0300], [iTC\_CC-SyAD-0316], [iTC\_CC-SyAD-1270], [iTC\_CC-SyAD-0292], [iTC\_CC\_ATP\_SwHA-0260], [iTC\_CC\_ATP\_SwHA-0263]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [PointVSLNotExceed](#PointVSLNotExceed) | √ | √ | BOOLEAN |
| [ZoneVSLNotExceed](#ZoneVSLNotExceed) | √ | √ | BOOLEAN |

## F46-Determine Over Energy

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [InhibitEmergencyBrake](#InhibitEmergencyBrake) | Internal | F71-Outputs to Rolling-stock |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [PointVSLNotExceed](#PointVSLNotExceed) | Internal | F45-Process Singularities |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [ZoneVSLNotExceed](#ZoneVSLNotExceed) | Internal | F45-Process Singularities |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [TrainEnergyControlDisabled](#TrainEnergyControlDisabled) | √ | √ | BOOLEAN |

### Processing

无论本周期的列车能量超过了车身范围内限制区域的能量限制，或是下游限制点的能量限制，ATP均应当触发EB，确保列车能满足线路上的速度或位置限制要求。

ATP shall trigger EB to make sure that the train can fulfill the limitation requirements of speed or location, no matter the train energy exceeded the energy of train intersected zone or constraint point of the downstream.

#### Motion protection inhibition

[iTC\_CC\_ATP-SwRS-0064]

MotionProtectionInhibition，表示ATP不负责列车位置的监控。其状态来自于项目可配置的列车输入采集。

def [MotionProtectionInhibition](#MotionProtectionInhibition)(k):

return [Offline.GetMotionProtectionInhibition](#GetMotionProtectionInhibition)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0218], [iTC\_CC-SyAD-0339], [iTC\_CC-SyAD-0343], [iTC\_CC-SyAD-1003], [iTC\_CC-SyAD-1285], [iTC\_CC-SyAD-1306], [iTC\_CC-SyAD-0802], [iTC\_CC\_ATP\_SwHA-0200]

[End]

#### Determing over energy

[iTC\_CC\_ATP-SwRS-0325]

TrainPossiblyInOverEnergy，列车能量大于限制点或限制区能量，即超能。

If the train energy exceeds the zone of point vital speed limitation, ATP shall consider the train possibly over energy.

def [TrainPossiblyInOverEnergy](#TrainPossiblyInOverEnergy)(k):

return (not [ZoneVSLNotExceed](#ZoneVSLNotExceed)(k)

or not [PointVSLNotExceed](#PointVSLNotExceed)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0316], [iTC\_CC-SyAD-0227], [iTC\_CC-SyAD-0276], [iTC\_CC-SyAD-0137]

[End]

[iTC\_CC\_ATP-SwRS-0326]

TrainEnergyControlDisabled，在RM模式下不报超能。

If the RMF or RMR mode selected, ATP shall not monitor the train energy.

def [TrainEnergyControlDisabled](#TrainEnergyControlDisabled)(k):

return [MotionProtectionInhibition](#MotionProtectionInhibition)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0227], [iTC\_CC-SyAD-0344], [iTC\_CC-SyAD-0796], [iTC\_CC-SyAD-0913], [iTC\_CC-SyAD-1398]

[End]

[iTC\_CC\_ATP-SwRS-0327]

EBforOverEnergy，超能后是否输出EB

ATP shall request emergency braking if train is possibly in over-energy and train speed control enabled and if following conditions fulfilled:

* the train is not detected at filtered stop,
* or the train is detected at filtered stop and:
* safe immobilization customization setting for this control indicates to use emergency brake,
* or safe immobilization customization setting for this control indicates to use emergency brake when it was already applied.

def [EBforOverEnergy](#EBforOverEnergy)(k):

return ([**TrainPossiblyInOverEnergy**](#TrainPossiblyInOverEnergy)(k)

and not [TrainEnergyControlDisabled](#TrainEnergyControlDisabled)(k)

and (not [TrainFilteredStopped](#TrainFilteredStopped)(k)

or ([TrainFilteredStopped](#TrainFilteredStopped)(k)

and ([**ATPsetting**](#ATPsetting).MPauthImmoBehaviourAtFS

is **IB\_APPLY\_EMERGENCY\_BRAKE**)

or (([**ATPsetting**](#ATPsetting).MPauthImmoBehaviourAtFS

is **IB\_APPLY\_EMERGENCY\_BRAKE\_WHEN\_TRIGGERED**)

and not [InhibitEmergencyBrake](#InhibitEmergencyBrake)(k-1)))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0227], [iTC\_CC-SyAD-0276], [iTC\_CC-SyAD-0316], [iTC\_CC-SyAD-0317], [iTC\_CC\_ATP\_SwHA-0134], [iTC\_CC-SyAD-0137]

[End]

[iTC\_CC\_ATP-SwRS-0328]

PBforOverEnergy，超能停车后是否继续输出PB

ATP shall request parking braking if train is possibly in over-energy and train speed control enabled and if following conditions fulfilled:

* the train is detected at filtered stop,
* and safe immobilization customization setting for this control indicates to use parking brake.

def [PBforOverEnergy](#PBforOverEnergy)(k):

return ([**TrainPossiblyInOverEnergy**](#TrainPossiblyInOverEnergy)(k)

and not [TrainEnergyControlDisabled](#TrainEnergyControlDisabled)

and ([TrainFilteredStopped](#TrainFilteredStopped)(k)

and ([**ATPsetting**](#ATPsetting).MPauthImmoBehaviourAtFS is **IB\_APPLY\_PARKING\_BRAKE**)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0230], , [iTC\_CC-SyAD-0227], [iTC\_CC-SyAD-0231], [iTC\_CC-SyAD-0317], [iTC\_CC\_ATP\_SwHA-0183]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [EBforOverEnergy](#EBforOverEnergy) | √ | √ | BOOLEAN |
| [MotionProtectionInhibition](#MotionProtectionInhibition) | √ | √ | BOOLEAN |
| [PBforOverEnergy](#PBforOverEnergy) | √ | √ | BOOLEAN |
| [TrainPossiblyInOverEnergy](#TrainPossiblyInOverEnergy) | √ | √ | BOOLEAN |

## F5-Monitor Train Position and Speed

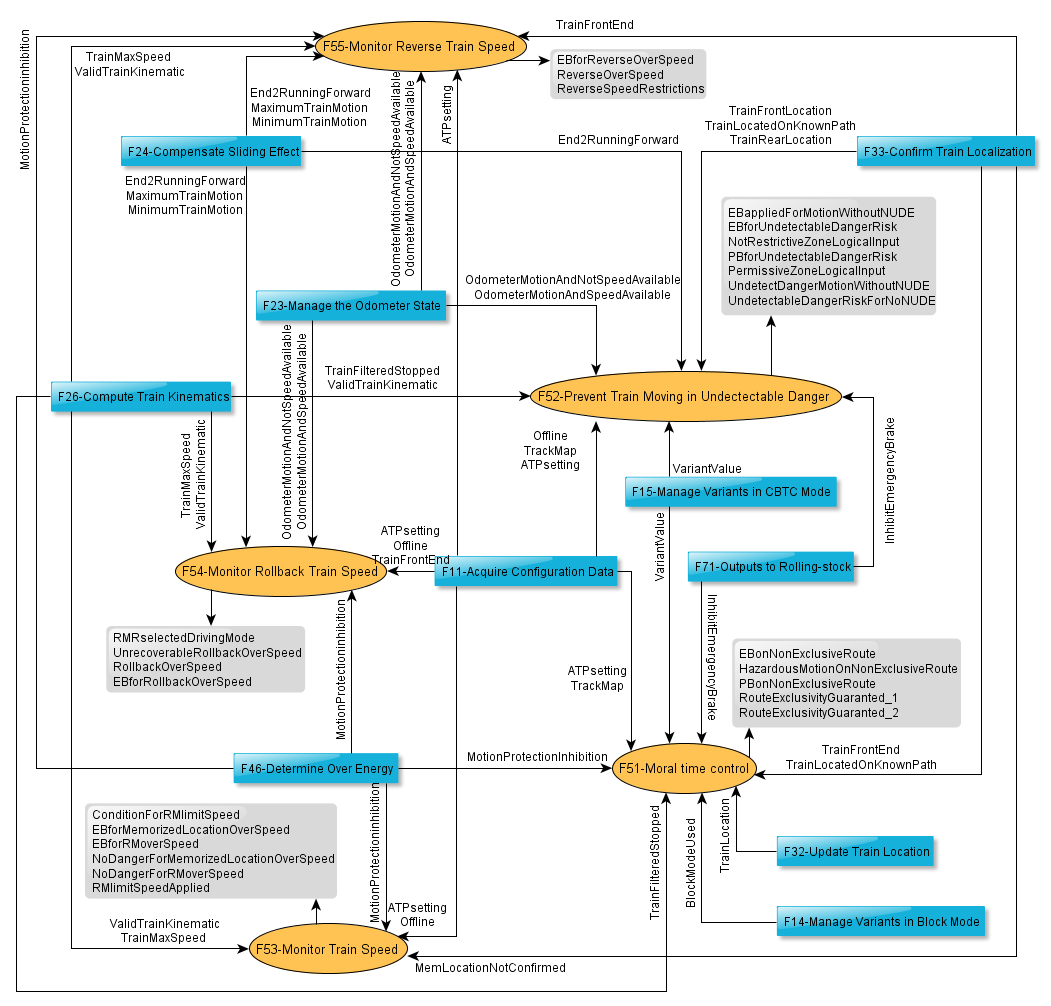


Figure 5‑19 SART modeling of function F5

## F51-Moral Time Control

本模块用于监控列车是否在模糊时间监控区停留过长时间。

This module monitors whether the train is staying in the fuzzy time area longer than expected.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [BlockModeUsed](#BlockModeUsed) | Internal | F14-Manage Variants in Block Mode |
| [InhibitEmergencyBrake](#InhibitEmergencyBrake) | Internal | F71-Outputs to Rolling-stock |
| [MotionProtectionInhibition](#MotionProtectionInhibition) | Internal | F46-Determine Over Energy |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainFrontEnd](#TrainFrontEnd) | Internal | F33-Confirm Train Localization |
| [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath) | Internal | F33-Confirm Train Localization |
| [TrainLocation](#TrainLocation) | Internal | F32-Update Train Location |
| [VariantValue](#VariantValue) | Internal | F15-Manage Variants in CBTC Mode |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [NotOnRestrictiveMoralTimeArea\_1](#NotOnRestrictiveMoralTimeArea_1) | √ | √ | BOOLEAN |
| [NotOnRestrictiveMoralTimeArea\_2](#NotOnRestrictiveMoralTimeArea_2) | √ | √ | BOOLEAN |

### Processing

由于ATP对车头最大定位进行过估，使得出现列车实际未通过某信号机，但ATP已认为通过了该信号机的状态。在此情况下，如果联锁取消该信号机下游进路，而ATP由于已通过该信号机从而不监控其状态变化，仍能授权列车继续前进，从而可能导致危险的发生。

Because ATP always overestimates the maximum location of the train front end, it is possible to consider the train has crossed a signal but actually not. In this case, if the CI cancelled the route downstream of this signal and it changed to restricted status accordingly, ATP cannot prohibit the train to moving on the unauthorized route (because ATP think the train has crossed the signal and the CI cannot change the route).

为避免上述情形，在Block运营模式下，ATP需增加考虑信号机下游的一段可配置区域，如果车头最大最小定位与该区域有交集时，仍需监控信号机状态，并确保在信号机变为限制状态后的一段可配置时间内列车通过了该段区域，否则将触发EB。在CBTC模式下，模糊区监控由ZC负责。

To avoid this situation, in Block operation mode, ATP needs to consider an additional area downstream the signal, called moral time area. If the train location intersects with the moral time area more than a configurable period just after the signal change from the permissive to the restrictive status, ATP shall trigger the EB. In CBTC mode, the monitoring of moral time is handled by ZC.

[iTC\_CC\_ATP-SwRS-0292]

NotOnRestrictiveMoralTimeArea\_1，当列车定位时，ATP需判断**END\_1**端车头的内外侧 定位是否与该端车头朝向的“限制状态”信号机下游的模糊时间区有无交集。其中模糊时间区定义为信号机下游长度为[ATPsetting](#ATPsetting).MTdistance的一段范围。

当满足下列所有条件时，设置[NotOnRestrictiveMoralTimeArea\_1](#NotOnRestrictiveMoralTimeArea_1)为**True**：

* 列车已确认定位；
* 并且：
* **END\_1**端车头的内外侧定位与**END\_1**端车头朝向的信号机下游模糊区没有交集；
* 或者，**END\_1**端车头的内外侧定位与**END\_1**端车头朝向的信号机下游模糊区有交集，但该信号机是允许状态。

否则，应设置[NotOnRestrictiveMoralTimeArea\_1](#NotOnRestrictiveMoralTimeArea_1)为**False**。

def [NotOnRestrictiveMoralTimeArea\_1](#NotOnRestrictiveMoralTimeArea_1)(k):

Signal = [TrackMap.ExistSingularityInReverseZone](#ExistSingularityInReverseZone)(**SGL\_SIGNAL**,

[TrainLocation](#TrainLocation).Ext1,

[ATPsetting](#ATPsetting).MTdistance + [TrainLocation](#TrainLocation)(k).Uncertainty)

return ([TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k)

and (Signal is **None**

or not Signal.BmMoralTime

or [VariantValue](#VariantValue)(Signal.Variant, k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0321], [iTC\_CC-SyAD-0322], [iTC\_CC-SyAD-1295], [iTC\_CC\_ATP\_SwHA-0116]

[End]

[iTC\_CC\_ATP-SwRS-0293]

NotOnRestrictiveMoralTimeArea\_2，当列车定位时，ATP需判断**END\_2**端车头的内外侧定位是否与该端车头朝向的“限制状态”信号机下游的模糊时间区有无交集。

当满足下列所有条件时，设置[NotOnRestrictiveMoralTimeArea\_2](#NotOnRestrictiveMoralTimeArea_2)为**True**：

* 列车已确认定位；
* 并且：
* **END\_2**端车头的内外侧定位与**END\_2**端车头朝向的信号机下游模糊区没有交集；
* 或者，**END\_2**端车头的内外侧定位与**END\_2**端车头朝向的信号机下游模糊区有交集，但该信号机是允许状态。

否则，应设置[NotOnRestrictiveMoralTimeArea\_2](#NotOnRestrictiveMoralTimeArea_2)为**False**。

def [NotOnRestrictiveMoralTimeArea\_2](#NotOnRestrictiveMoralTimeArea_2)(k):

Signal = [TrackMap.ExistSingularityInReverseZone](#ExistSingularityInReverseZone)(**SGL\_SIGNAL**,

[TrainLocation](#TrainLocation).Ext2,

[ATPsetting](#ATPsetting).MTdistance + [TrainLocation](#TrainLocation)(k).Uncertainty)

return ([TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k)

and (Signal is **None**

or not Signal.BmMoralTime

or [VariantValue](#VariantValue)(Signal.Variant, k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0321], [iTC\_CC-SyAD-0322], [iTC\_CC-SyAD-1295], [iTC\_CC\_ATP\_SwHA-0116]

[End]

[iTC\_CC\_ATP-SwRS-0294]

RouteExclusivityGuaranted\_1，如果列车在车头1对应方向且限制状态的模糊时间区内超过项目设定时间，则ATP应将该值设为限制状态。其中MoralTimeTimer\_1为记录列车在车头1对应方向的限制状态模糊时间区内的时间。

If ATP cannot determine train is [NotOnRestrictiveMoralTimeArea\_1](#NotOnRestrictiveMoralTimeArea_1), and if this situation lasts more than [ATPsetting](#ATPsetting). MTtimeout cycles, ATP shall consider that route exclusivity is not guaranteed and [RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1) shall be set to **False**.

If ATP detects that train is [NotOnRestrictiveMoralTimeArea\_1](#NotOnRestrictiveMoralTimeArea_1), route exclusivity shall consider as guaranteed for that direction of travel and [RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1) shall set to **True**

def [RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1)(k):

if (Initialization):

[RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1) = **False**

elif ([RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1)(k-1)

and not [NotOnRestrictiveMoralTimeArea\_1](#NotOnRestrictiveMoralTimeArea_1)(k)):

if (MoralTimeTimer\_1(k-1) < round.floor([ATPsetting](#ATPsetting).MTtimeout)):

MoralTimeTimer\_1 = MoralTimeTimer\_1(k-1) + 1

else:

[RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1) = **False**

elif ([NotOnRestrictiveMoralTimeArea\_1](#NotOnRestrictiveMoralTimeArea_1)(k)):

MoralTimeTimer\_1 = 1

[RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1) = **True**

else:

[RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1) = [RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1)(k-1)

return [RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0321], [iTC\_CC\_ATP\_SwHA-0118]

[End]

[iTC\_CC\_ATP-SwRS-0295]

RouteExclusivityGuaranted\_2，如果列车在车头2对应方向且限制状态的模糊时间区内超过项目设定时间，则ATP应将该值设置为限制状态，其中MoralTimeTimer\_2为记录列车在车头2对应方向限制状态模糊时间区内的时间。

If ATP cannot determine train is [NotOnRestrictiveMoralTimeArea\_2](#NotOnRestrictiveMoralTimeArea_2), and if this situation lasts more than [ATPsetting](#ATPsetting).MTtimeout cycles, ATP shall consider that route exclusivity is not guaranteed and [RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2) shall set to **False**.

If ATP detects that train is [NotOnRestrictiveMoralTimeArea\_2](#NotOnRestrictiveMoralTimeArea_2), route exclusivity shall consider as guaranteed for that direction of travel and [RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2) shall set to **True**

def [RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2)(k):

if (Initialization):

[RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2) = **False**

elif (RouteExclusivityGuaranted 2(k-1)

and not [NotOnRestrictiveMoralTimeArea\_2](#NotOnRestrictiveMoralTimeArea_2)(k)):

if (MoralTimeTimer\_2(k-1) < round.floor([ATPsetting](#ATPsetting).MTtimeout)):

MoralTimeTimer\_2 = MoralTimeTimer\_2(k-1) + 1

else:

[RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2) = **False**

elif ([NotOnRestrictiveMoralTimeArea\_2](#NotOnRestrictiveMoralTimeArea_2)(k)):

MoralTimeTimer\_2 = 1

[RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2) = **True**

else:

[RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2) = [RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2)(k-1)

return [RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0321], [iTC\_CC\_ATP\_SwHA-0118]

[End]

[iTC\_CC\_ATP-SwRS-0296]

HazardousMotionOnNonExclusiveRoute，非RM的BM模式下，如果列车在激活端车头方向的限制状态的Moral Time区停止超时预设时间，则ATP认为当前处于“非独占进路”的风险之中。

If [RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1) is **False**, ATP shall request emergency braking if and only if:

* [TrainFrontEnd](#TrainFrontEnd) is not **END\_2**,
* RM forward nor RM reverse are not selected,
* and block mode is not selected.

If [RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2) is **False**, ATP shall request emergency braking if and only if:

* [TrainFrontEnd](#TrainFrontEnd) is not **END\_1**,
* RM forward nor RM reverse driving mode are not selected,
* and block mode is not selected.

def [HazardousMotionOnNonExclusiveRoute](#HazardousMotionOnNonExclusiveRoute)(k):

return (((not [RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1)(k) and ([TrainFrontEnd](#TrainFrontEnd)(k)!= **END\_2**))

or (not [RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2)(k) and ([TrainFrontEnd](#TrainFrontEnd)(k)!= **END\_1**)))

and not [MotionProtectionInhibition](#MotionProtectionInhibition)(k)

and [BlockModeUsed](#BlockModeUsed)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0323], [iTC\_CC-SyAD-0324]

[End]

[iTC\_CC\_ATP-SwRS-0297]

PBonNonExclusiveRoute，当由于MoralTime监控导致的停车后，是否保持输出停车制动的取决于项目配置。

ATP shall request parking braking if train considered too near from a non-exclusive route and if following conditions are fulfilled:

* the train is detected at filtered stop,
* safe immobilization customization setting for this control indicates to use parking brake.

[PBonNonExclusiveRoute](#PBonNonExclusiveRoute)(k)

= [HazardousMotionOnNonExclusiveRoute](#HazardousMotionOnNonExclusiveRoute)(k)

and [TrainFilteredStopped](#TrainFilteredStopped)(k)

and ([ATPsetting](#ATPsetting).MTimmoBehaviourAtFS == **IB\_APPLY\_PARKING\_BRAKE**)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0324], [iTC\_CC-SyAD-0325], [iTC\_CC\_ATP\_SwHA-0117]

[End]

[iTC\_CC\_ATP-SwRS-0298]

EBonNonExclusiveRoute，如果当前处于“非独占进路”的风险中，且列车在移动，则ATP应当输出EB；如果当前已停车，则是否继续输出EB取决于项目配置。

ATP shall request emergency braking if train considered too near from a non-exclusive route and if following conditions are fulfilled:

* the train is not detected at filtered stop,
* or the train is detected at filtered stop and:
* safe immobilization customization setting for this control indicates to use emergency brake,
* or safe immobilization customization setting for this control indicates to use emergency brake when it was already applied.

[EBonNonExclusiveRoute](#EBonNonExclusiveRoute)(k)

= （[HazardousMotionOnNonExclusiveRoute](#HazardousMotionOnNonExclusiveRoute)(k)

and (not [TrainFilteredStopped](#TrainFilteredStopped)(k)

or ([TrainFilteredStopped](#TrainFilteredStopped)(k)

and (([ATPsetting](#ATPsetting).MTimmoBehaviourAtFS == **IB\_APPLY\_EMERGENCY\_BRAKE**)

or (([ATPsetting](#ATPsetting).MTimmoBehaviourAtFS == **IB\_APPLY\_EMERGENCY\_BRAKE\_WHEN\_TRIGGERED**)

and not [InhibitEmergencyBrake](#InhibitEmergencyBrake)(k-1))))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0324], [iTC\_CC-SyAD-0325], [iTC\_CC\_ATP\_SwHA-0119]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [EBonNonExclusiveRoute](#EBonNonExclusiveRoute) | √ | √ | BOOLEAN |
| [HazardousMotionOnNonExclusiveRoute](#HazardousMotionOnNonExclusiveRoute) | √ | √ | BOOLEAN |
| [PBonNonExclusiveRoute](#PBonNonExclusiveRoute) | √ | √ | BOOLEAN |
| [RouteExclusivityGuaranted\_1](#RouteExclusivityGuaranted_1) | √ | √ | BOOLEAN |
| [RouteExclusivityGuaranted\_2](#RouteExclusivityGuaranted_2) | √ | √ | BOOLEAN |

## F52-Prevent Train Moving in Undetectable Danger

本模块用于监控在没有NUDE的情况下列车是否运行超过了指定距离。

This function used to monitor whether the train is running with undetectable danger risk.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [End2RunningForward](#End2RunningForward) | Internal | F27-Compute Train Kinematics |
| [InhibitEmergencyBrake](#InhibitEmergencyBrake) | Internal | F71-Outputs to Rolling-stock |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainFrontLocation](#TrainFrontLocation) | Internal | F33-Confirm Train Localization |
| [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath) | Internal | F33-Confirm Train Localization |
| [TrainRearLocation](#TrainRearLocation) | Internal | F33-Confirm Train Localization |
| [ValidTrainKinematic](#ValidTrainKinematic) | Internal | F27-Compute Train Kinematics |
| [VariantValue](#VariantValue) | Internal | F15-Manage Variants in CBTC Mode |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [NUDEdistanceAccount\_1](#NUDEdistanceAccount_1) | √ | √ | NUMERIC\_32 |
| [NUDEdistanceAccount\_2](#NUDEdistanceAccount_2) | √ | √ | NUMERIC\_32 |
| [NoUndetectableDanger\_1](#NoUndetectableDanger_1) | √ | √ | BOOLEAN |
| [NoUndetectableDanger\_2](#NoUndetectableDanger_2) | √ | √ | BOOLEAN |

### Processing

#### Permissive/ not restrictive zone logical inputs

在项目中，可通过在线路地图上配置vital zone及其相应的“区域逻辑输入”来实现列车在该vital zone所表示的“允许区”和“限制区”的监控：

* 对于允许区，vital zone有“允许区逻辑输入”属性，也可能含有相关变量。ATP需监控列车是否完全被包含在该vital zone表示的“允许区”内：即列车定位完全在该区域内，且该区域没有相关变量或变量为允许状态。此时，ATP认为列车“完全在允许区内”。
* 对于限制区，vital zone有“非限制区逻辑输入”属性，也可能含有相关变量。ATP需监控列车没有任何部分在该“限制区”内：即列车定位与该区域没有交集；或者，与列车定位有交集的限制区存在变量，且该变量为允许状态。此时，ATP认为列车“与限制区没有交集”。

项目可配置，当列车“完全在允许区内”，或者“与限制区没有交集”时，ATP不认为当前处于“无法侦测的风险”中，即无需处理；反之，则认为当前处于“无法侦测的风险”，需要立即输出紧急制动。

项目可配的vital zone逻辑输入属性，如Table 5‑10所示。

Table 5‑10 Logical inputs for vital zone

|  |  |
| --- | --- |
| PzType for [PermissiveZoneLogicalInput](#PermissiveZoneLogicalInput) | NrzType for [NotRestrictiveZoneLogicalInput](#NotRestrictiveZoneLogicalInput) |
| **PermissiveZoneLogicalInput\_1** | **NotRestrictiveZoneLogicalInput\_1** |
| **PermissiveZoneLogicalInput\_2** | **NotRestrictiveZoneLogicalInput\_2** |
| **PermissiveZoneLogicalInput\_3** | **NotRestrictiveZoneLogicalInput\_3** |
| **PermissiveZoneLogicalInput\_4** | **NotRestrictiveZoneLogicalInput\_4** |

**NOTES：**

通过配置“允许区逻辑输入”，包括但不限于实现以下功能：

* 授权在指定区域内以ATB模式控车
* 授权在指定区域内进行长距离RMR倒车

[iTC\_CC\_ATP-SwRS-0717]

PermissiveZoneLogicalInput，允许区逻辑输入。

def [PermissiveZoneLogicalInput](#PermissiveZoneLogicalInput)(PzType, k):

return ([TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k)

and IncludedInVitalZone(PzType, k))

其中，列车定位完全包含在vital zone中的条件如下：

def IncludedInVitalZone(PzType, k):

for Vz in ([TrackMap](#TrackMap).[AllSingsBtwTwoLocs](#AllSingsBtwTwoLocs)(SGL\_VITAL\_ZONE,

[TrackMap](#TrackMap).[BlockOrigin](#BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrainFrontLocation](#TrainFrontLocation)(k).Max)):

if (Vz.[PermissiveZoneLogicalInput](#PermissiveZoneLogicalInput) is PzType

and [TrackMap](#TrackMap).[LocationInZone](#LocationInZone)([TrainFrontLocation](#TrainFrontLocation)(k).Max,

Vz.Location,

Vz.Length,

Vz.Orientation)

and [TrackMap](#TrackMap).[LocationInZone](#LocationInZone)([TrainRearLocation](#TrainRearLocation)(k).Min,

Vz.Location,

Vz.Length,

Vz.Orientation)

and (Vz.Variant is **None**

or ([VariantValue](#VariantValue)(Vz.Variant, k)))):

return **True**

else:

continue

return **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0971], [iTC\_CC-SyAD-1036], [iTC\_CC-SyAD-1299], [iTC\_CC-SyAD-1316], [iTC\_CC-SyAD-1318], [iTC\_CC-SyAD-1378], [iTC\_CC-SyAD-1394]

[End]

**NOTES：**

通过配置“非限制区逻辑输入”，包括但不限于实现以下功能：

* 禁止丢失BM信标；
* 禁止以ATB模式进入非PSD站台。

[iTC\_CC\_ATP-SwRS-0718]

NotRestrictiveZoneLogicalInput，非限制区逻辑输入。

def [NotRestrictiveZoneLogicalInput](#NotRestrictiveZoneLogicalInput)(NrzType, k):

return ([TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k)

and NotIntersectVitalZone(NrzType, k))

其中，列车与vital zone没有交集的判别条件如下：

def NotIntersectVitalZone(NrzType, k):

for Vz in ([TrackMap](#TrackMap).[AllSingsBtwTwoLocs](#AllSingsBtwTwoLocs)(SGL\_VITAL\_ZONE,

[TrackMap](#TrackMap).[BlockOrigin](#BlockOrigin)([TrainRearLocation](#TrainRearLocation)(k).Min),

[TrainFrontLocation](#TrainFrontLocation)(k).Max)):

if (Vz.[NotRestrictiveZoneLogicalInput](#NotRestrictiveZoneLogicalInput) is NrzType

and ([TrackMap](#TrackMap).[IntersectionOfTwoZones](#IntersectionOfTwoZones)([TrainRearLocation](#TrainRearLocation)(k).Min,

[TrainFrontLocation](#TrainFrontLocation)(k).Max,

vz.Location,

[TrackMap](#TrackMap).[CalculateZoneBorder](#CalculateZoneBorder)(vz.Location, vz.Length))

is not **None**)

and (Vz.Variant is **None**

or not [VariantValue](#VariantValue)(Vz.Variant, k))):

return **False**

else:

continue

return **True**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source= [iTC\_CC-SyAD-0200], [iTC\_CC-SyAD-1315], [iTC\_CC-SyAD-1317], [iTC\_CC-SyAD-1352]

[End]

#### No undetectable dange

在未建立ATB模式的情况下，ATP应当避免列车处于“无法侦测的危险”的风险中，即列车向司机未授权的方向运行。在此种情况下，如果列车运行超过项目配置的距离，则ATP应当触发EB，并保持一段可配置的时间。

ATP shall avoid the train in "undetectable danger risk", which means that the train runs on the direction not authorized by the driver or by CCNV in ATB driving mode. In this case, if the train runs more than a configurable distance, ATP shall trigger emergency brake and keep it for a configurable period.

[iTC\_CC\_ATP-SwRS-0582]

NoUndetectableDanger\_1，已监控向**END\_1**方向的运行，其状态来自于项目可配置的列车输入采集。

The No Undetectable Danger in Extremity 1 shall be consider as permissive status according to project configuration.

def [NoUndetectableDanger\_1](#NoUndetectableDanger_1)(k):

return [Offline](#Offline).[NoUndetectableDanger\_1](#NoUndetectableDanger_1)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source= [iTC\_CC-SyAD-0351], [iTC\_CC-SyAD-0971],[iTC\_CC-SyAD-1036], [iTC\_CC\_ATP\_SwHA-0215], [iTC\_CC-SyAD-1352], [iTC\_CC\_ATP\_SwHA-0265]

[End]

[iTC\_CC\_ATP-SwRS-0583]

NoUndetectableDanger\_2，其状态来自于项目可配置的列车输入采集。

The "No Undetectable Danger in Extremity 2" shall be consider as permissive status according to project configuration.

def [NoUndetectableDanger\_2](#NoUndetectableDanger_2)(k):

return [Offline](#Offline).[NoUndetectableDanger\_2](#NoUndetectableDanger_2)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source= [iTC\_CC-SyAD-0351], [iTC\_CC-SyAD-0971], [iTC\_CC-SyAD-1036], [iTC\_CC\_ATP\_SwHA-0215], [iTC\_CC-SyAD-1352], [iTC\_CC\_ATP\_SwHA-0265]

[End]

**NOTES：**

在离线数据中，可将PermissiveZoneLogicalInput或NotRestrictiveZoneLogicalInput作为配置数据[NoUndetectableDanger\_1](#NoUndetectableDanger_1)或[NoUndetectableDanger\_2](#NoUndetectableDanger_2)的输入变量参与运算，即结合线路上vital zone及其变量的设置，实现NUDE的监控。

[iTC\_CC\_ATP-SwRS-0285]

UndetectableDangerRiskForNoNUDE，当前两端车头都没有NUDE输入，则认为列车存在“无法侦测的风险”。

If there is neither No Undetectable Danger in Extremity 1 nor No Undetectable Danger in Extremity 2 inputs, ATP shall consider the train is possible under the risk of undetectable danger.

def [UndetectableDangerRiskForNoNUDE](#UndetectableDangerRiskForNoNUDE)(k):

return (not [NoUndetectableDanger\_1](#NoUndetectableDanger_1)(k)

and not [NoUndetectableDanger\_2](#NoUndetectableDanger_2)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0349], [iTC\_CC-SyAD-0350], [iTC\_CC-SyAD-0351], [iTC\_CC-SyAD-0354], [iTC\_CC-SyAD-1320], [iTC\_CC\_ATP\_SwHA-0215]

[End]

[iTC\_CC\_ATP-SwRS-0286]

PBforUndetectableDangerRisk，当停车且存在“无法侦测的风险”时，如果项目配置为输出停车制动，则ATP应当输出停车制动。

ATP shall request a parking braking if the possibility of an undetected danger has proven to be and if following conditions are fulfilled:

* the train is detected at filtered stop,
* safe immobilization customization setting for this control indicates to use parking brake.

[PBforUndetectableDangerRisk](#PBforUndetectableDangerRisk)(k)

= ([UndetectableDangerRiskForNoNUDE](#UndetectableDangerRiskForNoNUDE)(k)

and [TrainFilteredStopped](#TrainFilteredStopped)(k)

and ([ATPsetting](#ATPsetting).NUDEimmoBehaviourAtFS == **IB\_APPLY\_PARKING\_BRAKE**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0354], [iTC\_CC-SyAD-0355], [iTC\_CC\_ATP\_SwHA-0113]

[End]

[iTC\_CC\_ATP-SwRS-0287]

NUDEdistanceAccount\_1，监控当司机未授权向**END\_1**方向运行时，列车向**END\_1**方向运行的距离，该值为非负数，

* 若在初始化阶段，或NUDE1为**True**，或已经EB并停车，则等于0；
* 否则，当测速无效时，将其设置为默认值
* 否则，当里程计已初始化后：
* 如果[MaximumTrainMotion](#MaximumTrainMotion)大于0，则等于上周期累加距离加上本周期最大位移，最小取0。
* 而如果[MaximumTrainMotion](#MaximumTrainMotion)小于等于0，则使用上周期值加最小位移（实际上就是减小该累加值，倒车），最小取0
* 否则，保持累加距离不变。

When the driver does not authorize the train running toward the **END\_1**, ATP shall accumulate the distance of the train running toward to the **END\_1**.

* If in initialization, or the [NoUndetectableDanger\_1](#NoUndetectableDanger_1) is **True**, or the train has triggered EB and has stopped, ATP set this distance to 0;
* Else if train kinematic has invalid, ATP set this distance to the default value.
* Else if the odometer has initialized:
* If the [MaximumTrainMotion](#MaximumTrainMotion)is greater than 0, ATP accumulate the maximum movement in this cycle with the distance of last cycle;
* Or if the [MaximumTrainMotion](#MaximumTrainMotion)is less than or equal to 0, ATP use the minimum movement of this cycle plus to the distance last cycle (in fact, decrease the accumulated distance). The minimum of this accumulated distance is 0.
* Otherwise, keep the distance unchanged.

def [NUDEdistanceAccount\_1](#NUDEdistanceAccount_1)(k):

if (INTIALIZATION

or [NoUndetectableDanger\_1](#NoUndetectableDanger_1)(k)

or ([EBappliedForMotionWithoutNUDE](#EBappliedForMotionWithoutNUDE)(k-1) and [TrainFilteredStopped](#TrainFilteredStopped)(k)))

return 0

elif ([ValidTrainKinematic](#ValidTrainKinematic)(k) != **True**)

return [ATPsetting](#ATPsetting).NUDEdistanceWithoutMotionAvailable

elif ([OdometerState](#OdometerState)(k) is **INITIALIZED**)

if ([End1RunningForward](#End1RunningForward)(k))

return [max](#max)(0, [NUDEdistanceAccount\_1](#NUDEdistanceAccount_1)(k-1)+ [MaximumTrainMotion](#MaximumTrainMotion)(k))

else:

return [max](#max)(0, [NUDEdistanceAccount\_1](#NUDEdistanceAccount_1)(k-1)+ [MinimumTrainMotion](#MinimumTrainMotion)(k))

else:

return [NUDEdistanceAccount\_1](#NUDEdistanceAccount_1)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0352], [iTC\_CC-SyAD-0353], [iTC\_CC-SyAD-1319], [iTC\_CC\_ATP\_SwHA-0216]

[End]

[iTC\_CC\_ATP-SwRS-0288]

NUDEdistanceAccount\_2，监控当司机未授权向**END\_2**方向运行时，列车向**END\_2**方向运行的距离，该值为非正数，

* 若在初始化阶段，或NUDE2为**True**，或已经EB并停车，则等于0；
* 否则，当测速无效时，将其设置为默认值；
* 否则，当里程计已经初始化后：
* 若 [MaximumTrainMotion](#MaximumTrainMotion)小于0，则等于上周期累加距离加上本周期最大位移，最大取0。
* 若[MaximumTrainMotion](#MaximumTrainMotion)大于等于0，则使用上周期值加最小位移，最大取0。
* 否则，保持累加距离不变。

When the driver does not authorize the train running toward the **END\_2**, ATP shall accumulate the distance of the train running toward to the **END\_2**.

* If in initialization, or the [NoUndetectableDanger\_2](#NoUndetectableDanger_2) is **True**, or the train has triggered EB and has stopped, ATP set this distance to 0;
* Else if train kinematic has invalid, ATP set this distance to the default value.
* Else if the odometer has initialized:
* if the [MaximumTrainMotion](#MaximumTrainMotion) is less than 0, ATP accumulate the maximum movement in this cycle with the distance of last cycle;
* Else: if the [MaximumTrainMotion](#MaximumTrainMotion) is greater than or equal to 0, ATP use the minimum movement of this cycle plus to the distance last cycle (in fact, decrease the accumulated distance). The minimum of this accumulated distance is 0.
* Otherwise, keep the distance unchanged.

def [NUDEdistanceAccount\_2](#NUDEdistanceAccount_2)(k):

if (INTIALIZATION

or [NoUndetectableDanger\_2](#NoUndetectableDanger_2)(k)

or ([EBappliedForMotionWithoutNUDE](#EBappliedForMotionWithoutNUDE)(k-1) and [TrainFilteredStopped](#TrainFilteredStopped)(k)))

return 0

elif ([ValidTrainKinematic](#ValidTrainKinematic)(k) != **True**)

return -1 \* [ATPsetting](#ATPsetting).NUDEdistanceWithoutMotionAvailable(k)

elif ([OdometerState](#OdometerState)(k) is **INITIALIZED**)

if ([End2RunningForward](#End2RunningForward)(k))

return [min](#min)(0, [NUDEdistanceAccount\_2](#NUDEdistanceAccount_2)(k-1)+ [MaximumTrainMotion](#MaximumTrainMotion)(k))

else:

return [min](#min)(0, [NUDEdistanceAccount\_2](#NUDEdistanceAccount_2)(k-1)+ [MinimumTrainMotion](#MinimumTrainMotion)(k))

else:

return [NUDEdistanceAccount\_2](#NUDEdistanceAccount_2)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0352], [iTC\_CC-SyAD-0353], [iTC\_CC-SyAD-1319], [iTC\_CC\_ATP\_SwHA-0216]

[End]

[iTC\_CC\_ATP-SwRS-0289]

UndetectDangerMotionWithoutNUDE，列车运行超过限定距离，但仍有车头未检测到NUDE。

When the train has moved without NUDE more than project-restricted distance, ATP shall set this value to **True**.

[UndetectDangerMotionWithoutNUDE](#UndetectDangerMotionWithoutNUDE)(k)

= (not [NoUndetectableDanger\_1](#NoUndetectableDanger_1)(k)

and ([NUDEdistanceAccount\_1](#NUDEdistanceAccount_1)(k)> [ATPsetting](#ATPsetting).NUDElimitDistance))

or (not [NoUndetectableDanger\_2](#NoUndetectableDanger_2)(k)

and ([NUDEdistanceAccount\_2](#NUDEdistanceAccount_2)(k)< -1 \* [ATPsetting](#ATPsetting).NUDElimitDistance)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0352], [iTC\_CC\_ATP\_SwHA-0114]

[End]

[iTC\_CC\_ATP-SwRS-0290]

EBappliedForMotionWithoutNUDE，保证由NUDE导致的EB会延迟一段时间。即：

* 当[UndetectDangerMotionWithoutNUDE](#UndetectDangerMotionWithoutNUDE)为**True**时，设置EBappliedForMotionWithoutNUDE为**True**；
* 当[UndetectDangerMotionWithoutNUDE](#UndetectDangerMotionWithoutNUDE)由**True**变为**False**后，还需保持EBappliedForMotionWithoutNUDE 在[ATPsetting](#ATPsetting).NUDEtrainStopDurationBeforeEBrelease时间内为**True**；
* 超过上述时间后，该值为**False**。

The EB request shall be maintained to **True** during the application time [ATPsetting](#ATPsetting).NUDEtrainStopDurationBeforeEBrelease, if the train has moved without NUDE more than project restricted distance.

* When [UndetectDangerMotionWithoutNUDE](#UndetectDangerMotionWithoutNUDE) is **True**, ATP shall set [EBappliedForMotionWithoutNUDE](#EBappliedForMotionWithoutNUDE) to **True**;
* When [UndetectDangerMotionWithoutNUDE](#UndetectDangerMotionWithoutNUDE) change from **True** to **False**, ATP shall maintain [EBappliedForMotionWithoutNUDE](#EBappliedForMotionWithoutNUDE) to **True** in period [ATPsetting](#ATPsetting).NUDEtrainStopDurationBeforeEBrelease；
* Over the time, set this value to **False**.

def [EBappliedForMotionWithoutNUDE](#EBappliedForMotionWithoutNUDE)(k):

if ([UndetectDangerMotionWithoutNUDE](#UndetectDangerMotionWithoutNUDE)(k)):

NudeEBreleaseCounter = 0

return **True**

elif (NudeEBreleaseCounter < [ATPsettings](#ATPsettings).NUDEtrainStopDurationBeforeEBrelease):

NudeEBreleaseCounter = NudeEBreleaseCounter(k-1) + 1

return **True**

else:

return **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0352], [iTC\_CC-SyAD-0355], [iTC\_CC-SyAD-1320], [iTC\_CC\_ATP\_SwHA-0114]

[End]

[iTC\_CC\_ATP-SwRS-0291]

EBforUndetectableDangerRisk，由“无法侦测的危险”导致EB并停车后，ATP应当根据项目配置判断是否输出EB。

When the train has triggered emergency brake causing by the "undetectable danger risk" and has stopped, ATP shall determine whether keeping the EB output according to the project configuration.

[EBforUndetectableDangerRisk](#EBforUndetectableDangerRisk)(k)

= ([EBappliedForMotionWithoutNUDE](#EBappliedForMotionWithoutNUDE)(k)

or ([UndetectableDangerRiskForNoNUDE](#UndetectableDangerRiskForNoNUDE)(k)

and (not [TrainFilteredStopped](#TrainFilteredStopped)(k)

or ([ATPsetting](#ATPsetting).NUDEimmoBehaviourAtFS == **IB\_APPLY\_EMERGENCY\_BRAKE**)

or (([ATPsetting](#ATPsetting).NUDEimmoBehaviourAtFS == **IB\_APPLY\_EMERGENCY\_BRAKE\_WHEN\_TRIGGERED**)

and (not [InhibitEmergencyBrake](#InhibitEmergencyBrake)(k-1))))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0352], [iTC\_CC-SyAD-0355], [iTC\_CC-SyAD-1320], [iTC\_CC\_ATP\_SwHA-0115]

[End]

### Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Safety Critical | | Observable | | Logical Type | |
| [EBappliedForMotionWithoutNUDE](#EBappliedForMotionWithoutNUDE) | | √ | | √ | | BOOLEAN |
| [EBforUndetectableDangerRisk](#EBforUndetectableDangerRisk) | | √ | | √ | | BOOLEAN |
| [NotRestrictiveZoneLogicalInput](#NotRestrictiveZoneLogicalInput) | | √ | | √ | | BOOLEAN |
| [PBforUndetectableDangerRisk](#PBforUndetectableDangerRisk) | | √ | | √ | | BOOLEAN |
| [PermissiveZoneLogicalInput](#PermissiveZoneLogicalInput) | | √ | | √ | | BOOLEAN |
| [UndetectDangerMotionWithoutNUDE](#UndetectDangerMotionWithoutNUDE) | | √ | | √ | | BOOLEAN |
| [UndetectableDangerRiskForNoNUDE](#UndetectableDangerRiskForNoNUDE) | | √ | | √ | | BOOLEAN |

## F53-Monitor Train Speed

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [MemLocationNotConfirmed](#MemLocationNotConfirmed) | Internal | F33-Confirm Train Localization |
| [MotionProtectioninhibition](#MotionProtectioninhibition) | Internal | F46-Determine Over Energy |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [TrainMaxSpeed](#TrainMaxSpeed) | Internal | F27-Compute Train Kinematics |
| [ValidTrainKinematic](#ValidTrainKinematic) | Internal | F27-Compute Train Kinematics |

### Processing

#### RM mode speed monitoring

如果ATP检测到司机选择了限制人工（RM）模式，则根据项目配置的RM模式限速值监控列车是否超速。

If the RM forward or reverse modes selected, ATP shall monitor whether the train is overspeed based on the RM limitation.

[iTC\_CC\_ATP-SwRS-0743]

ConditionForRMlimitSpeed，当前应用哪种RM限速。ATP最多支持项目配置**MAX\_RM\_CONDITION\_NB**种RM限速。

def [ConditionForRMlimitSpeed](#ConditionForRMlimitSpeed)(i, k):

return [Offline.GetConditionForRMlimitSpeed](#GetConditionForRMlimitSpeed)(i, k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1265], [iTC\_CC-SyAD-1309]

[End]

[iTC\_CC\_ATP-SwRS-0744]

RMlimitSpeedApplied，根据列车输入，判断当前应当监控的RM限速

def [RMlimitSpeedApplied](#RMlimitSpeedApplied)(k):

for i in [range](#range)(0, **MAX\_RM\_CONDITION\_NB**):

if ([ConditionForRMlimitSpeed](#ConditionForRMlimitSpeed)(i, k)):

return [ATPsetting](#ATPsetting).MPinhibitionLimitSpeed[i]

else:

continue

return 0

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0278], [iTC\_CC-SyAD-1309], [iTC\_CC-SyAD-1394]

[End]

**NOTES:**

在配置数据中，[ConditionForRMlimitSpeed](#ConditionForRMlimitSpeed)必须配置列车的“各种模式”及其相应的“最大”限速。其原因是防止ATP未发现相应模式的限速时，将[RMlimitSpeedApplied](#RMlimitSpeedApplied)设置为0，影响可用性。但对于这些限速的监控，仅在[MotionProtectionInhibition](#MotionProtectionInhibition)时实施。

[iTC\_CC\_ATP-SwRS-0497]

NoDangerForRMoverSpeed，列车速度小于等于RM模式下的限速。

ATP estimates that current train maximum speed not exceeds the RM limit speed.

def [NoDangerForRMoverSpeed](#NoDangerForRMoverSpeed)(k):

return ([ValidTrainKinematic](#ValidTrainKinematic)(k)

and [TrainMaxSpeed](#TrainMaxSpeed)(k) <= [RMlimitSpeedApplied](#RMlimitSpeedApplied)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0275], [iTC\_CC-SyAD-0278], [iTC\_CC-SyAD-0364], [iTC\_CC-SyAD-1394], [iTC\_CC\_ATP\_SwHA-0121]

[End]

[iTC\_CC\_ATP-SwRS-0734]

EBforRMoverSpeed，若在RM模式下，列车速度大于RM模式限速，则将输出EB。

def [EBforRMoverSpeed](#EBforRMoverSpeed)(k):

return (not [NoDangerforRMoverSpeed](#NoDangerforRMoverSpeed)(k)

and [MotionProtectionInhibition](#MotionProtectionInhibition)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0275], [iTC\_CC-SyAD-0278], [iTC\_CC-SyAD-0364], [iTC\_CC-SyAD-1394], [iTC\_CC\_ATP\_SwHA-0121]

[End]

#### Monitoring memorized location speed limit

[iTC\_CC\_ATP-SwRS-0751]

NoDangerForMemorizedLocationOverSpeed，在使用记忆定位而还未读到确认信标时，ATP监控列车速度是否超过项目限制值。

def [NoDangerForMemorizedLocationOverSpeed](#NoDangerForMemorizedLocationOverSpeed)(k):

return (not [MemLocationNotConfirmed](#MemLocationNotConfirmed)(k)

or [[TrainMaxSpeed](#TrainMaxSpeed)](#MaximumTrainSpeed) < [ATPsetting](#ATPsetting).MemLocLimitSpeed)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1272], [iTC\_CC\_ATP\_SwHA-0266]

[End]

[iTC\_CC\_ATP-SwRS-0719]

EBforMemorizedLocationOverSpeed，在使用记忆定位而还未读到确认信标时，ATP应确保列车速度不超过项目限制值。

def [EBforMemorizedLocationOverSpeed](#EBforMemorizedLocationOverSpeed)(k):

return (not [NoDangerforMemorizedLocationOverSpeed](#NoDangerforMemorizedLocationOverSpeed)(k)

and not [MotionProtectionInhibition](#MotionProtectionInhibition)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1272], [iTC\_CC\_ATP\_SwHA-0266]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [ConditionForRMlimitSpeed](#ConditionForRMlimitSpeed) | √ | √ | BOOLEAN |
| [EBforMemorizedLocationOverSpeed](#EBforMemorizedLocationOverSpeed) | √ | √ | BOOLEAN |
| [EBforRMoverSpeed](#EBforRMoverSpeed) | √ | √ | BOOLEAN |
| [NoDangerForMemorizedLocationOverSpeed](#NoDangerForMemorizedLocationOverSpeed) | √ | √ | BOOLEAN |
| [NoDangerForRMoverSpeed](#NoDangerForRMoverSpeed) | √ | √ | BOOLEAN |
| [RMlimitSpeedApplied](#RMlimitSpeedApplied) | √ | √ | BOOLEAN |

## F54-Monitor Rollback Train Speed

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [End2RunningForward](#End2RunningForward) | Internal | F27-Compute Train Kinematics |
| [MaximumTrainMotion](#MaximumTrainMotion) | Internal | F27-Compute Train Kinematics |
| [MinimumTrainMotion](#MinimumTrainMotion) | Internal | F27-Compute Train Kinematics |
| [MotionProtectionInhibition](#MotionProtectionInhibition) | Internal | F46-Determine Over Energy |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [TrainFrontEnd](#TrainFrontEnd) | Internal | F11-Acquire Configuration Data |
| [TrainMaxSpeed](#TrainMaxSpeed) | Internal | F27-Compute Train Kinematics |
| [ValidTrainKinematic](#ValidTrainKinematic) | Internal | F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [RollbackDistanceAccount\_1](#RollbackDistanceAccount_1) | √ | √ | NUMERIC\_32 |
| [RollbackDistanceAccount\_2](#RollbackDistanceAccount_2) | √ | √ | NUMERIC\_32 |

### Processing

[iTC\_CC\_ATP-SwRS-0065]

RMRselectedDrivingMode，是否选择了RMR倒车模式。其状态来自于项目可配置的列车输入采集。

[RMRselectedDrivingMode](#RMRselectedDrivingMode) represents the choice of RMR.

def [RMRselectedDrivingMode](#RMRselectedDrivingMode)(k):

return [Offline](#Offline).[GetRMRselectedDrivingMode](#GetRMRselectedDrivingMode)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0218], [iTC\_CC-SyAD-0340], [iTC\_CC-SyAD-0344], [iTC\_CC-SyAD-1003], [iTC\_CC-SyAD-1308], [iTC\_CC\_ATP\_SwHA-0201]

[End]

列车在非RMR模式下，向激活车头的反向运行，称之为回溜。ATP监控回溜的速度必须满足项目配置的允许速度，否则将触发EB。如果回溜的距离超过项目配置的最大距离时，ATP应当触发无法缓解的永久EB。

When the train is not on RMR mode, and the train moved backward related to the active cab, called rollback. ATP shall request EB if the speed of the rollback is greater than the project limits. If the rollback distance is greater than the project limits, ATP shall request the permanent EB, which cannot release.

[iTC\_CC\_ATP-SwRS-0300]

RollbackDistanceAccount\_1，累计回溜的距离（负值表示在回溜）：

* 初始化时设置该值为0；
* 否则，如果列车运动学无效，则设置为配置参数的默认值；
* 否则，在**END\_1**激活且未选择RMR模式的前提下：
* 若里程计已初始化，且列车向**END\_1**方向运行，则累加最小位移，若超过0则取0，否则是一个负值。
* 否则，若里程计齿数齿号匹配，则累加列车最大位移
* 否则，即里程计未初始化，则保持累计距离不变。
* 其他情况，保持累计距离不变。

When train front extremity is **END\_1** and traction effort is supposed to be in the direction of travel, [RollbackDistanceAccount\_1](#RollbackDistanceAccount_1) is the estimated maximum distance which separates current front extremity 1 position to last most forward position reached by this extremity. ATP shall evaluate [RollbackDistanceAccount\_1](#RollbackDistanceAccount_1) in order to control that speed does not exceed [ATPsetting](#ATPsetting).MPnotAuthLimitSpeed .

def [RollbackDistanceAccount\_1](#RollbackDistanceAccount_1)(k):

if (Initialization)

return 0

elif (not [ValidTrainKinematic](#ValidTrainKinematic)(k)):

return (-1 \* [ATPsetting](#ATPsetting).MPnotAuthDistWithoutMotionAvailable)

elif ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**

and not [RMRselectedDrivingMode](#RMRselectedDrivingMode)(k)):

if ([OdometerState](#OdometerState)(k) is **INITIALIZED**):

if ([End1RunningForward](#End1RunningForward)(k)):

return [min](#min)(0, [RollbackDistanceAccount\_1](#RollbackDistanceAccount_1)(k-1) + [MinimumTrainMotion](#MinimumTrainMotion)(k))

else:

return ([RollbackDistanceAccount\_1](#RollbackDistanceAccount_1)(k-1) + [MaximumTrainMotion](#MaximumTrainMotion)(k))

else:

return [RollbackDistanceAccount\_1](#RollbackDistanceAccount_1)(k-1)

else:

return [RollbackDistanceAccount\_1](#RollbackDistanceAccount_1)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0326], [iTC\_CC-SyAD-0328]

[End]

[iTC\_CC\_ATP-SwRS-0301]

RollbackDistanceAccount\_2，累计回溜的距离（负值表示在回溜）：

* 初始化时设置该值为0；
* 否则，如果列车运动学无效，则设置为配置参数的默认值；
* 否则，在**END\_2**激活且未选择RMR模式的前提下：
* 若里程计已初始化，且列车向**END\_2**方向运行，则减去最小位移，若超过0则取0，否则是一个负值。
* 否则，若里程计已初始化，则减去列车最大位移
* 否则，即里程计还未初始化，则保持累计距离不变。
* 其他情况，保持累计距离不变。

When train front extremity is **END\_2** and traction effort is supposed to be in the direction of travel, [RollbackDistanceAccount\_2](#RollbackDistanceAccount_2) is the estimated maximum distance which separates current front extremity 2 position to last most forward position reached by this extremity. ATP shall evaluate [RollbackDistanceAccount\_2](#RollbackDistanceAccount_2) in order to control that speed does not exceed [ATPsetting](#ATPsetting).MPnotAuthLimitSpeed.

def [RollbackDistanceAccount\_2](#RollbackDistanceAccount_2)(k):

if (Initialization)

return 0

elif (not [ValidTrainKinematic](#ValidTrainKinematic)(k)):

return (-1 \* [ATPsetting](#ATPsetting).MPnotAuthDistWithoutMotionAvailable)

elif ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_2**

and not [RMRselectedDrivingMode](#RMRselectedDrivingMode)(k)):

if ([OdometerState](#OdometerState)(k) is **INITIALIZED**):

if ([End2RunningForward](#End2RunningForward)(k)):

return [min](#min)(0, [RollbackDistanceAccount\_2](#RollbackDistanceAccount_2)(k-1) - [MinimumTrainMotion](#MinimumTrainMotion)(k))

else:

return ([RollbackDistanceAccount\_2](#RollbackDistanceAccount_2)(k-1) — [MaximumTrainMotion](#MaximumTrainMotion)(k))

else:

return [RollbackDistanceAccount\_2](#RollbackDistanceAccount_2)(k-1)

else:

return [RollbackDistanceAccount\_2](#RollbackDistanceAccount_2)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0326], [iTC\_CC-SyAD-0328]

[End]

[iTC\_CC\_ATP-SwRS-0302]

UnrecoverableRollbackOverSpeed，如果ATP检测到列车已经回退超过项目限制的最大距离，则设置永久回退超速

From ATP power-up, [UnrecoverableRollbackOverSpeed](#UnrecoverableRollbackOverSpeed) shall initialize to **False**.

[UnrecoverableRollbackOverSpeed](#UnrecoverableRollbackOverSpeed) shall be set to **True** if and only if following conditions are fulfilled:

* driving selector indicates that traction effort is supposed to be in the direction of travel,
* train front extremity is **END\_2** or **END\_1**,
* and rollback limit speed currently applicable is null for this direction of travel.

Once [UnrecoverableRollbackOverSpeed](#UnrecoverableRollbackOverSpeed) set as **True**, it shall stay at state **True** while ATP is not reboot.

def [UnrecoverableRollbackOverSpeed](#UnrecoverableRollbackOverSpeed)(k):

return ([UnrecoverableRollbackOverSpeed](#UnrecoverableRollbackOverSpeed)(k-1)

or (not [RMRselectedDrivingMode](#RMRselectedDrivingMode)(k)

and (([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_2**

and ([abs](#abs)([RollbackDistanceAccount\_2](#RollbackDistanceAccount_2)(k))

> [ATPsetting](#ATPsetting).MPnotAuthLimitDistance))

or ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**

and ([abs](#abs)([RollbackDistanceAccount\_1](#RollbackDistanceAccount_1)(k))

> [ATPsetting](#ATPsetting).MPnotAuthLimitDistance)))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0275], [iTC\_CC-SyAD-0328], [iTC\_CC-SyAD-0364], [iTC\_CC\_ATP\_SwHA-0123]

[End]

[iTC\_CC\_ATP-SwRS-0303]

RollbackOverSpeed，下列任一条件满足，认为回退超速

* 若车头2激活，位移为**END\_1**方向，未选择RMR模式
* 车速大于当前回退距离所在限速
* 若车头1激活，位移为**END\_2**方向，未选择RMR模式
* 车速大于当前回退距离所在限速
* 列车运动学无效
* 已发生了永久回退超速错误

[RollbackOverSpeed](#RollbackOverSpeed) shall be **True** if and only if following conditions are fulfilled:

* driving selector indicates that traction effort is supposed to be in the direction of travel,
* train front extremity is **END\_2** or **END\_1**,
* and movement observed is in the opposite direction of travel,
* and over-estimated train speed is greater than [ATPsetting](#ATPsetting).MPnotAuthLimitSpeed currently applicable for this direction of travel and rollback speed restrictions is not null.

Or:

* train has reached a position due a rollback movement which is unrecoverable,

Or:

* train kinematic is invalid,

def [RollbackOverSpeed](#RollbackOverSpeed)(k):

return (not [ValidTrainKinematic](#ValidTrainKinematic)(k)

or [UnrecoverableRollbackOverSpeed](#UnrecoverableRollbackOverSpeed)(k)

or (not [RMRselectedDrivingMode](#RMRselectedDrivingMode)(k)

and ([TrainMaxSpeed](#TrainMaxSpeed)(k) >= [ATPsetting](#ATPsetting).MPnotAuthLimitSpeed

and (([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_2**

and [End1RunningForward](#End1RunningForward)(k) and not [End2RunningForward](#End2RunningForward)(k))

or ([TrainFrontEnd](#TrainFrontEnd) is **END\_1**

and [End2RunningForward](#End2RunningForward)(k) and not [End1RunningForward](#End1RunningForward)(k))))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0275], [iTC\_CC-SyAD-0326], [iTC\_CC-SyAD-0329], [iTC\_CC-SyAD-0364], [iTC\_CC\_ATP\_SwHA-0122]

[End]

[iTC\_CC\_ATP-SwRS-0304]

EBforRollbackOverSpeed，如果ATP检测到回溜超速，则输出EB

ATP shall request emergency braking if a reverse speed limit is over-run for unwilling rollback or excessive reverse motion.

[EBforRollbackOverSpeed](#EBforRollbackOverSpeed) = [RollbackOverSpeed](#RollbackOverSpeed)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0328], [iTC\_CC-SyAD-0329], [iTC\_CC-SyAD-0364], [iTC\_CC\_ATP\_SwHA-0124]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [RMRselectedDrivingMode](#RMRselectedDrivingMode) | √ | √ | BOOLEAN |
| [UnrecoverableRollbackOverSpeed](#UnrecoverableRollbackOverSpeed) | √ | √ | BOOLEAN |
| [RollbackOverSpeed](#RollbackOverSpeed) | √ | √ | BOOLEAN |
| [EBforRollbackOverSpeed](#EBforRollbackOverSpeed) | √ | √ | BOOLEAN |

## F55-Monitor Reverse Train Speed

### Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Nature | | Provided by | |
| [ATPsetting](#ATPsetting) | | Internal | | F11-Acquire Configuration Data |
| [End2RunningForward](#End2RunningForward) | | Internal | | F27-Compute Train Kinematics |
| [MaximumTrainMotion](#MaximumTrainMotion) | | Internal | | F27-Compute Train Kinematics |
| [MinimumTrainMotion](#MinimumTrainMotion) | | Internal | | F27-Compute Train Kinematics |
| [MotionProtectionInhibition](#MotionProtectionInhibition) | | Internal | | F46-Determine Over Energy |
| [RMRselectedDrivingMode](#RMRselectedDrivingMode) | | Internal | | F54-Monitor Rollback Train Speed |
| [TrainFrontEnd](#TrainFrontEnd) | | Internal | | F33-Confirm Train Localization |
| [TrainMaxSpeed](#TrainMaxSpeed) | | Internal | | F27-Compute Train Kinematics |
| [ValidTrainKinematic](#ValidTrainKinematic) | | Internal | | F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [ReverseDistanceAccount\_1](#ReverseDistanceAccount_1) | √ | √ | NUMERIC\_32 |
| [ReverseDistanceAccount\_2](#ReverseDistanceAccount_2) | √ | √ | NUMERIC\_32 |

### Processing

当列车在RMR模式下向激活车头的反向运行时，ATP需监控列车速度不能超过项目配置的反向运行最大速度，如果超过该速度，则触发EB。如果列车已RMR反向运行超过项目配置的最大距离，则ATP保持输出EB，禁止列车继续已RMR模式反向运行。

When the train selected on RMR mode, the train moved reversely related to the active cab. ATP shall request EB if the speed of the reverse is greater than the project limits. If the train reversing distance is greater than the project limits, ATP shall keep requesting EB to inhibit train moving on RMR mode.

[iTC\_CC\_ATP-SwRS-0759]

LongDistanceReverseAuthorized，长距离倒车模式是否授权，其状态来自于项目可配置的列车输入采集。

[LongDistanceReverseAuthorized](#LongDistanceReverseAuthorized) represents the authorization of long distance reverse.

def [LongDistanceReverseAuthorized](#LongDistanceReverseAuthorized)(k):

return [Offline.GetLongDistanceReverseAuthorized](#GetLongDistanceReverseAuthorized)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1394], [iTC\_CC\_ATP\_SwHA-0278], [iTC\_CC-SyAD-1442]

[End]

**NOTES：**

对于在指定区域内“长距离倒车”功能的监控，是通过离线数据配置，由需求SwRS-0717和0743实现的：

* 在项目的线路地图中：
* 在需要进行长距离倒车的区域配置VitalZone及其相应的[PermissiveZoneLogicalInput](#PermissiveZoneLogicalInput)属性，不需要变量。ATP保证 当列车定位完全在该区域内 时，设置 [PermissiveZoneLogicalInput](#PermissiveZoneLogicalInput)为**True**
* 在项目的安全配置参数中：
* 配置[LongDistanceReverseAuthorized](#LongDistanceReverseAuthorized)为上述[PermissiveZoneLogicalInput](#PermissiveZoneLogicalInput)与CCNV授权长距离倒车（即CCNV判断过停）同时为**True**，此时ATP不再监控普通的RMR倒车距离，也不监控Rollback；
* 在TrainType/MotionProtection/inhibition/LimitSpeed中，配置一个选择RMR模式且 [LongDistanceReverseAuthorized](#LongDistanceReverseAuthorized)为**True**的限速，作为长距离RMR的限速。ATP保证在该条件满足时，以该限速监控列车。

ATP对于Rollback，RMR，长距离倒车（LDR）三种距离的累加条件如表所示：

Table 5‑11 Backward distance account rules

|  |  |  |
| --- | --- | --- |
|  | [LongDistanceReverseAuthorized](#LongDistanceReverseAuthorized) | not [LongDistanceReverseAuthorized](#LongDistanceReverseAuthorized) |
| [RMRselectedDrivingMode](#RMRselectedDrivingMode) | RMR向前溜车：  不抵消RMR距离，不抵消Rollback距离 | RMR向前溜车：  抵消RMR距离，不抵消Rollback距离 |
| 长距离倒车：  不累加RMR距离，不累加Rollback距离 | 普通RMR倒车：  累加RMR距离，不累加Rollback距离 |
| not [RMRselectedDrivingMode](#RMRselectedDrivingMode) | LDR区域向前：  不抵消RMR距离，抵消Rollback距离 | 正常向前：  抵消RMR距离，抵消Rollback距离 |
| LDR区域回溜：  不累加RMR距离，累加Rollback距离 | 普通回溜：  不累加RMR距离，累加Rollback距离 |

[iTC\_CC\_ATP-SwRS-0305]

ReverseDistanceAccount\_1，累加RMR模式下的倒车距离（负值表示倒车）：

* 初始化时设置该值为0；
* 否则，如果列车运动学无效，则设置为配置参数的默认值；
* 否则，在**END\_1**激活且非长距离倒车授权的前提下：
* 若里程计已初始化，且列车向**END\_1**方向运行，则减小倒车距离绝对值，大于零则等于0
* 否则，如果里程计已初始化，且选择RMR模式，则累加倒车距离
* 否则，即里程计还未初始化，则保持距离不变。
* 其他情况，保持累计距离不变。

When train front extremity is **END\_1** and traction effort is supposed to be in the opposite direction of travel, [ReverseDistanceAccount\_1](#ReverseDistanceAccount_1) is the estimated maximum distance which separates current front extremity 1 position to last most forward position reached by this extremity. ATP shall evaluate [ReverseDistanceAccount\_1](#ReverseDistanceAccount_1) in order to control that speed does not exceed reverse speed limit function.

def [ReverseDistanceAccount\_1](#ReverseDistanceAccount_1)(k):

if (Initialization)

return 0

elif (not [ValidTrainKinematic](#ValidTrainKinematic)(k)):

return [ATPsetting](#ATPsetting).ReverseDistWithoutMotionAvailable

elif ([TrainFrontEnd](#TrainFrontEnd) is **END\_1**

and not [LongDistanceReverseAuthorized](#LongDistanceReverseAuthorized)(k)):

if ([OdometerState](#OdometerState)(k) is **INITIALIZED**):

if ([End1RunningForward](#End1RunningForward)(k)):

return [min](#min)(0, [ReverseDistanceAccount\_1](#ReverseDistanceAccount_1)(k-1) + [MinimumTrainMotion](#MinimumTrainMotion)(k))

elif ([RMRselectedDrivingMode](#RMRselectedDrivingMode)(k)):

return ([ReverseDistanceAccount\_1](#ReverseDistanceAccount_1)(k-1) + [MaximumTrainMotion](#MaximumTrainMotion)(k))

else:

return [ReverseDistanceAccount\_1](#ReverseDistanceAccount_1)(k-1)

else:

return [ReverseDistanceAccount\_1](#ReverseDistanceAccount_1)(k-1)

else:

return [ReverseDistanceAccount\_1](#ReverseDistanceAccount_1)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0331], [iTC\_CC-SyAD-1308]

[End]

[iTC\_CC\_ATP-SwRS-0306]

ReverseDistanceAccount\_2，累加RMR模式下的倒车距离（负值表示倒车）：

* 初始化时设置该值为0；
* 否则，如果列车运动学无效，则设置为配置参数的默认值；
* 否则，在**END\_2**激活且非长距离倒车授权的前提下：
* 如果里程计已初始化，且列车向**END\_2**方向运行，则减小倒车距离绝对值，大于零则等于0
* 否则，如果里程计已初始化，且选择RMR模式时累加倒车距离
* 否则，即里程计未初始化，则保持距离不变；
* 其他情况，保持累计距离不变。

When train front extremity is **END\_2** and traction effort is supposed to be in the opposite direction of travel, [ReverseDistanceAccount\_2](#ReverseDistanceAccount_2) is the estimated maximum distance which separates current front extremity 2 position to last most forward position reached by this extremity. ATP shall evaluate [ReverseDistanceAccount\_2](#ReverseDistanceAccount_2) in order to control that speed does not exceed [ReverseSpeedRestrictions](#ReverseSpeedRestrictions) reverse speed limit function.

def [ReverseDistanceAccount\_2](#ReverseDistanceAccount_2)(k):

if (Initialization)

return 0

elif (not [ValidTrainKinematic](#ValidTrainKinematic)(k)):

return [ATPsetting](#ATPsetting).ReverseDistWithoutMotionAvailable

elif ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_2**

and not [LongDistanceReverseAuthorized](#LongDistanceReverseAuthorized)(k)):

if ([OdometerState](#OdometerState)(k) is **INITIALIZED**):

if ([End2RunningForward](#End2RunningForward)(k))

return [min](#min)(0, [ReverseDistanceAccount\_2](#ReverseDistanceAccount_2)(k-1) - [MinimumTrainMotion](#MinimumTrainMotion)(k))

elif ([RMRselectedDrivingMode](#RMRselectedDrivingMode)(k)):

return ([ReverseDistanceAccount\_2](#ReverseDistanceAccount_2)(k-1) — [MaximumTrainMotion](#MaximumTrainMotion)(k))

else:

return [ReverseDistanceAccount\_2](#ReverseDistanceAccount_2)(k-1)

else:

return [ReverseDistanceAccount\_2](#ReverseDistanceAccount_2)(k-1)

else:

return [ReverseDistanceAccount\_2](#ReverseDistanceAccount_2)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0331], [iTC\_CC-SyAD-1308]

[End]

[iTC\_CC\_ATP-SwRS-0753]

ReverseSpeedRestrictions，根据当前计算的倒车累加距离在[ATPsetting](#ATPsetting).ReverseLimit数组中索引的当前最大允许倒车速度。

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0275], [iTC\_CC-SyAD-0330]

[End]

[iTC\_CC\_ATP-SwRS-0307]

ReverseOverSpeed，超过RMR模式限速的条件：

[ReverseOverSpeed](#ReverseOverSpeed) shall be **True** if following conditions fulfilled:

* driving selector indicates that traction effort is supposed to be in the opposite direction of travel,
* train front extremity is **END\_2** or **END\_1**,
* and movement observed is the opposite direction of travel,
* and:
* over-estimated train speed is greater than reverse speed restrictions currently applicable for this direction of travel,
* or else: if reverse speed restrictions currently applicable is null for this direction of travel,
* Or else: train kinematic is invalid.

def [ReverseOverSpeed](#ReverseOverSpeed)(k):

if (not [RMRselectedDrivingMode](#RMRselectedDrivingMode)(k)

or [LongDistanceReverseAuthorized](#LongDistanceReverseAuthorized)(k)):

return **False**

elif (not [ValidTrainKinematic](#ValidTrainKinematic)(k)):

return **True**

else:

return (([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_2**):

and (([End1RunningForward](#End1RunningForward)(k) and not [End2RunningForward](#End2RunningForward)(k)

and ([TrainMaxSpeed](#TrainMaxSpeed)(k)

> [ReverseSpeedRestrictions](#ReverseSpeedRestrictions)([ReverseDistanceAccount\_2](#ReverseDistanceAccount_2)(k))))

or ([ReverseSpeedRestrictions](#ReverseSpeedRestrictions)([ReverseDistanceAccount\_2](#ReverseDistanceAccount_2)(k)) == 0)))

or ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**

and (([End2RunningForward](#End2RunningForward)(k) and not [End1RunningForward](#End1RunningForward)(k)

and ([TrainMaxSpeed](#TrainMaxSpeed)(k) > [ReverseSpeedRestrictions](#ReverseSpeedRestrictions)([ReverseDistanceAccount\_1](#ReverseDistanceAccount_1)(k))))

or ([ReverseSpeedRestrictions](#ReverseSpeedRestrictions)([ReverseDistanceAccount\_1](#ReverseDistanceAccount_1)(k)) == 0))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0275], [iTC\_CC-SyAD-0330], [iTC\_CC-SyAD-0279], [iTC\_CC\_ATP\_SwHA-0125], [iTC\_CC-SyAD-1308]

[End]

[iTC\_CC\_ATP-SwRS-0308]

EBforReverseOverSpeed，由于RMR下倒车超速而导致EB

ATP shall request emergency braking if a reverse speed limit is over-run for unwilling rollback or excessive reverse motion.

[EBforReverseOverSpeed](#EBforReverseOverSpeed) = [ReverseOverSpeed](#ReverseOverSpeed)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0331], [iTC\_CC-SyAD-0279], [iTC\_CC-SyAD-1308], [iTC\_CC\_ATP\_SwHA-0126]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [EBforReverseOverSpeed](#EBforReverseOverSpeed) | √ | √ | BOOLEAN |
| [ReverseOverSpeed](#ReverseOverSpeed) | √ | √ | BOOLEAN |
| [ReverseSpeedRestrictions](#ReverseSpeedRestrictions) | √ | √ | NUMERIC\_32 |

## F6-Protect Passengers Entrance and Exit from the Train

本部分用于描述和车门、屏蔽门相关的需求。

This module describes the requirements for passengers entrance and exist of the train.

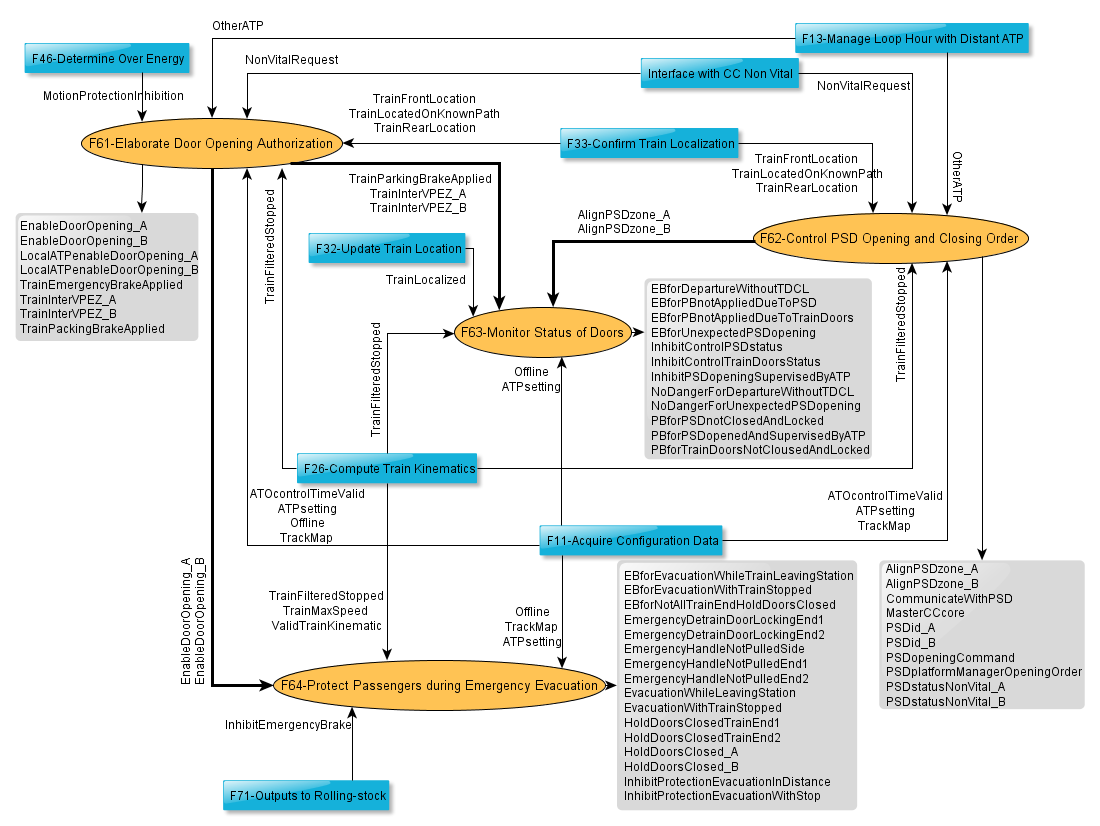


Figure 5‑20 SART modeling of function F6

## F61-Elaborate Door Opening Authorization

本模块用于判断授权开门的条件。

This module is judging the criteria for door opening.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATOcontrolTimeValid](#ATOcontrolTimeValid) | Internal | F11-Acquire Configuration Data |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [MotionProtectionInhibition](#MotionProtectionInhibition) | Internal | F46-Determine Over Energy |
| [NonVitalRequest](#NonVitalRequest) | External | Interface with CC Non Vital |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [OtherATP](#OtherATP) | Internal | F13-Manage Loop Hour with Distant ATP |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainFrontLocation](#TrainFrontLocation) | Internal | F33-Confirm Train Localization |
| [TrainFrontOrientation](#TrainFrontOrientation) | Internal | F33-Confirm Train Localization |
| [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath) | Internal | F33-Confirm Train Localization |
| [TrainRearLocation](#TrainRearLocation) | Internal | F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [NoVitalCorrectlyDocked](#NoVitalCorrectlyDocked) | × | √ | BOOLEAN |
| [TrainIncludedInVPEZ\_A](#TrainIncludedInVPEZ_A) | √ | √ | BOOLEAN |
| [TrainIncludedInVPEZ\_B](#TrainIncludedInVPEZ_B) | √ | √ | BOOLEAN |
| [TrainSafelyImmobilised](#TrainSafelyImmobilised) | √ | √ | BOOLEAN |
| [VPEZid\_A](#VPEZid_A) | √ | √ | NUMERIC\_32 |
| [VPEZid\_B](#VPEZid_B) | √ | √ | NUMERIC\_32 |

### Processing

ATP当且仅当确定列车车身范围完全在站台范围内，且保证安全停车时，才授权允许列车开站台指定侧的车门。

Only when the ATP has sure that the train is completely located in the platform area and the safely immobilized, ATP shall allow the train to open the specific door in the side of platform.

#### Vital passenger exchange zone

对于车站（VPEZ），PSD或紧急逃生区，在线路地图中的开门方向（即站台方向）均是按照上行方向来描述的，其属性有**LEFT**，**RIGHT**或**BOTH\_SIDES**。即如果列车向上行方向运营，则开门侧与线路地图中描述的方向相同；而若列车向下行方向运营，则开门侧应与线路地图中的方向相反。对于有**LEFT**属性的VPEZ，上行列车应开左侧车门，而下行列车应开右侧车门。

The **LEFT** or **RIGHT**, defined as attributes of VPEZ, PSD or EVAC singularities in track map, means the platform is on the left of right side of the train seen as towards the **UP** orientation.

但对于列车而言，其开门侧的是通过与车辆接口的A/B侧来对应的。其接口规则为：在列车上，以面对**END\_1**方向的左侧为A侧，右侧为B侧。那么，ATP就需要根据线路地图中的站台方向，列车车头最大定位的运营方向，以及激活的车头，与车辆接口的A/B侧建立对应关系，如Table 5‑12所示。

Table 5‑12 Convertion from platform side to train side

|  |  |  |  |
| --- | --- | --- | --- |
| [TrainFrontEnd](#TrainFrontEnd) | [TrainFrontOrientation](#TrainFrontOrientation) | Platform side in [TrackMap](#TrackMap) | Train door side |
| **END\_1** | **UP** | **LEFT** | SideA |
| **RIGHT** | SideB |
| **BOTH\_SIDES** | SideA, SideB |
| **DOWN** | **LEFT** | SideB |
| **RIGHT** | SideA |
| **BOTH\_SIDES** | SideA, SideB |
| **END\_2** | **UP** | **LEFT** | SideB |
| **RIGHT** | SideA |
| **BOTH\_SIDES** | SideA, SideB |
| **DOWN** | **LEFT** | SideA |
| **RIGHT** | SideB |
| **BOTH\_SIDES** | SideA, SideB |

[iTC\_CC\_ATP-SwRS-0262]

TrainInterVPEZ\_A，列车定位与A侧站台有交集；

* ATP初始化或者失位时，默认设置[TrainInterVPEZ\_A](#TrainInterVPEZ_A)为**False**；
* 当车身定位（即从车尾最小定位到车头最大定位）与站台区域（线路地图[TrackMap](#TrackMap)中由一对方向相反的**SGL\_VPEZ**奇点组成）有交集，且根据Table 5‑12判断为A侧时，设置[TrainInterVPEZ\_A](#TrainInterVPEZ_A)为**True**。
* 其他情况，设置[TrainInterVPEZ\_A](#TrainInterVPEZ_A)为**False**。

ATP shall determine whether there are intersection between the range of train locations and the vital passenger exchange zone of the side-A:

* In initialization or train delocalization, set the [TrainInterVPEZ\_A](#TrainInterVPEZ_A) as **False**;
* If there are intersection between the range of train locations (from the minimum train tail to the maximum train head) and the vital passenger exchange zone (composing by a pair of **SGL\_VPEZ** singularities with opposite direction in the train map), and the train door side A correspond to the platform according with Table 5‑12, set [TrainInterVPEZ\_A](#TrainInterVPEZ_A) as **True**:
* Otherwise, set [TrainInterVPEZ\_A](#TrainInterVPEZ_A) as **False**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0228], [iTC\_CC\_ATP\_SwHA-0104], [iTC\_CC\_ATP\_SwHA-0105], [iTC\_CC\_ATP\_SwHA-0106]

[End]

[iTC\_CC\_ATP-SwRS-0263]

TrainInterVPEZ\_B，列车定位与B侧站台有交集；

* ATP初始化或者失位时，默认设置[TrainInterVPEZ\_B](#TrainInterVPEZ_B)为**False**；
* 当车身定位（即从车尾最小定位到车头最大定位）与站台区域（线路地图[TrackMap](#TrackMap)中由一对方向相反的**SGL\_VPEZ**奇点组成）有交集，且根据Table 5‑12判断为B侧时，设置[TrainInterVPEZ\_B](#TrainInterVPEZ_B)为**True**。
* 其他情况，设置[TrainInterVPEZ\_B](#TrainInterVPEZ_B)为**False**。

ATP shall determine whether there are intersection between the range of train locations and the vital passenger exchange zone of the side-B:

* In initialization or train delocalization, set the [TrainInterVPEZ\_B](#TrainInterVPEZ_B) as **False**;
* If there are intersection between the range of train locations (from the minimum train tail to the maximum train head) and the vital passenger exchange zone (composing by a pair of **SGL\_VPEZ** singularities with opposite direction in the train map), and the train door side B correspond to the platform according with Table 5‑12, set [TrainInterVPEZ\_B](#TrainInterVPEZ_B) as **True**:
* Otherwise, set [TrainInterVPEZ\_B](#TrainInterVPEZ_B) as **False**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0228], [iTC\_CC\_ATP\_SwHA-0104], [iTC\_CC\_ATP\_SwHA-0105], [iTC\_CC\_ATP\_SwHA-0106]

[End]

[iTC\_CC\_ATP-SwRS-0560]

VPEZid\_A，与列车定位相交的A侧VPEZ的编号。

* 如果[TrainInterVPEZ\_A](#TrainInterVPEZ_A)为**True**，则记录该VPEZ的编号；
* 否则，设置为**None**。

[VPEZid\_A](#VPEZid_A) records the ID of the VPEZ, which intersected with the location of train on side A.

* If the [TrainInterVPEZ\_A](#TrainInterVPEZ_A), set as the ID of this VPEZ.
* Otherwise, set as **None**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0228]

[End]

[iTC\_CC\_ATP-SwRS-0561]

VPEZid\_B，与列车定位相交的B侧VPEZ的编号。

* 如果[TrainInterVPEZ\_B](#TrainInterVPEZ_B)为**True**，则记录该VPEZ的编号。；
* 否则，设置为**None**。

[VPEZid\_B](#VPEZid_B) records the ID of the VPEZ, which intersected with the location of train on side B.

* If the [TrainInterVPEZ\_B](#TrainInterVPEZ_B), then set as the ID of this VPEZ.
* Otherwise, set as **None**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0228]

[End]

[iTC\_CC\_ATP-SwRS-0264]

TrainIncludedInVPEZ\_A，列车车身是否完全在A侧VPEZ区域内。

* 初始化或者失位时，设置[TrainIncludedInVPEZ\_A](#TrainIncludedInVPEZ_A)为**False**；
* 如果列车车身（即从车尾最小定位到车头最大定位）完全在[VPEZid\_A](#VPEZid_A)范围中，且[TrainInterVPEZ\_A](#TrainInterVPEZ_A)为**True**，则设置[TrainIncludedInVPEZ\_A](#TrainIncludedInVPEZ_A)为**True**；
* 否则，设置[TrainIncludedInVPEZ\_A](#TrainIncludedInVPEZ_A)为**False**.

[TrainIncludedInVPEZ\_A](#TrainIncludedInVPEZ_A), ATP shall determine whether the whole train location belongs to one vital passenger exchange zone [VPEZid\_A](#VPEZid_A).

* In initialization or train delocalization, set the [TrainIncludedInVPEZ\_A](#TrainIncludedInVPEZ_A) as **False**;
* If the range of train locations (from the minimum train tail to the maximum train head) belongs to a VPEZ, and the [TrainInterVPEZ\_A](#TrainInterVPEZ_A) is **True**, then ATP shall set [TrainIncludedInVPEZ\_A](#TrainIncludedInVPEZ_A) as **True**;
* Otherwise, set [TrainIncludedInVPEZ\_A](#TrainIncludedInVPEZ_A) as **False**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0228], [iTC\_CC\_ATP\_SwHA-0106], [iTC\_CC\_ATP\_SwHA-0267]

[End]

[iTC\_CC\_ATP-SwRS-0265]

TrainIncludedInVPEZ\_B，列车车身是否完全在B侧VPEZ区域内。

* 初始化或者失位时，设置[TrainIncludedInVPEZ\_B](#TrainIncludedInVPEZ_B)为**False**；
* 如果列车车身（即从车尾最小定位到车头最大定位）完全在[VPEZid\_B](#VPEZid_B)范围中，且该[TrainInterVPEZ\_B](#TrainInterVPEZ_B)为**True**，则设置[TrainIncludedInVPEZ\_B](#TrainIncludedInVPEZ_B)为**True**；
* 否则，设置[TrainIncludedInVPEZ\_B](#TrainIncludedInVPEZ_B)为**False**。

[TrainIncludedInVPEZ\_B](#TrainIncludedInVPEZ_B), ATP shall determine whether the whole train location belongs to one vital passenger exchange zone [VPEZid\_B](#VPEZid_B).

* In initialization or train delocalization, set the [TrainIncludedInVPEZ\_B](#TrainIncludedInVPEZ_B) as **False**;
* If the range of train locations (from the minimum train tail to the maximum train head) belongs to a VPEZ, and the [TrainInterVPEZ\_B](#TrainInterVPEZ_B) is **True**, then ATP shall set [TrainIncludedInVPEZ\_B](#TrainIncludedInVPEZ_B) as **True**;
* Otherwise, set [TrainIncludedInVPEZ\_B](#TrainIncludedInVPEZ_B) as **False**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0228], [iTC\_CC\_ATP\_SwHA-0106], [iTC\_CC\_ATP\_SwHA-0267]

[End]

#### Safety immobilized

ATP需知道列车当前是否施加了制动，用于判断安全停稳信息。

The ATP needs to know whether the train has applied the parking or emergency brake, which used to judge the safe immobilization.

[iTC\_CC\_ATP-SwRS-0576]

TrainEmergencyBrakeApplied，列车是否施加了紧急制动。其状态来自于项目可配置的列车输入采集。

[TrainEmergencyBrakeApplied](#TrainEmergencyBrakeApplied) shows that whether the train has applied emergency brake.

* If the project is not configured, ATP shall consider the emergency brake has not applied by the train.
* Otherwise, if either of the end is in emergency brake, ATP considers the emergency brake has applied.

def [TrainEmergencyBrakeApplied](#TrainEmergencyBrakeApplied)(k):

return [Offline.GetTrainEmergencyBrakeApplied](#GetTrainEmergencyBrakeApplied)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source= [iTC\_CC-SyAD-1003], [iTC\_CC-SyAD-1224], [iTC\_CC\_ATP\_SwHA-0222]

[End]

[iTC\_CC\_ATP-SwRS-0073]

TrainParkingBrakeApplied，任一端车头已施加停车制动，则认为停车制动已施加。其状态来自于项目可配置的列车输入采集。

The term [TrainParkingBrakeApplied](#TrainParkingBrakeApplied) stands for that either of the train ends is in parking brake.

def [TrainParkingBrakeApplied](#TrainParkingBrakeApplied)(k):

return [Offline.GetTrainParkingBrakeApplied](#GetTrainParkingBrakeApplied)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0233], [iTC\_CC-SyAD-0234], [iTC\_CC-SyAD-0235], [iTC\_CC-SyAD-1003], [iTC\_CC-SyAD-1223], [iTC\_CC\_ATP\_SwHA-0222]

[End]

[iTC\_CC\_ATP-SwRS-0329]

TrainSafelyImmobilised，判断是否已经安全停车

ATP shall consider that train safely immobilized if:

* Train brake has detected safely applied, or train parking brake is detected;
* And train is detected at filtered stop.

def [TrainSafelyImmobilised](#TrainSafelyImmobilised)(k):

return (([TrainEmergencyBrakeApplied](#TrainEmergencyBrakeApplied)(k)

or [TrainParkingBrakeApplied](#TrainParkingBrakeApplied)(k))

and [TrainFilteredStopped](#TrainFilteredStopped)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0236], [iTC\_CC\_ATP\_SwHA-0135]

[End]

#### Door opening authorization

[iTC\_CC\_ATP-SwRS-0737]

NoVitalCorrectlyDocked，CCNV判断列车是否停在开门授权区内

def [NoVitalCorrectlyDocked](#NoVitalCorrectlyDocked)(k):

return ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k)

and [NonVitalRequest](#NonVitalRequest)(k).TrainInCorrectlyDockedZone)

#Category= Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0068], [iTC\_CC-SyAD-0229]

[End]

[iTC\_CC\_ATP-SwRS-0330]

LocalATPenableDoorOpening\_A，本ATP是否在站内允许开A侧车门：

ATP shall determine whether train doors on side A opening. The rules are following:

def [LocalATPenableDoorOpening\_A](#LocalATPenableDoorOpening_A)(k):

return ([TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k)

and [TrainIncludedInVPEZ\_A](#TrainIncludedInVPEZ_A)(k)

and [NoVitalCorrectlyDocked](#NoVitalCorrectlyDocked)(k)

and [TrainSafelyImmobilised](#TrainSafelyImmobilised)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0216], [iTC\_CC-SyAD-0229], [iTC\_CC-SyAD-0240], [iTC\_CC-SyAD-0241], [iTC\_CC-SyAD-0242], [iTC\_CC-SyAD-1225], [iTC\_CC\_ATP\_SwHA-0136]

[End]

[iTC\_CC\_ATP-SwRS-0331]

LocalATPenableDoorOpening\_B，本ATP是否授权开启B侧车门：

ATP shall determine whether train doors on side B opening. The rules are following:

def [LocalATPenableDoorOpening\_B](#LocalATPenableDoorOpening_B)(k):

return ([TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)(k)

and [TrainIncludedInVPEZ\_B](#TrainIncludedInVPEZ_B)(k)

and [NoVitalCorrectlyDocked](#NoVitalCorrectlyDocked)(k)

and [TrainSafelyImmobilised](#TrainSafelyImmobilised)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0216], [iTC\_CC-SyAD-0229], [iTC\_CC-SyAD-0240], [iTC\_CC-SyAD-0241], [iTC\_CC-SyAD-0242], [iTC\_CC-SyAD-1225], [iTC\_CC\_ATP\_SwHA-0136]

[End]

[iTC\_CC\_ATP-SwRS-0720]

EnableDoorOpening\_A，结合远端ATP结果的开门授权信息。

def [EnableDoorOpening\_A](#EnableDoorOpening_A)(k):

return ([LocalATPenableDoorOpening\_A](#LocalATPenableDoorOpening_A)(k)

or [OtherATP](#OtherATP).EnableDoorOpening\_A)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0242]

[End]

[iTC\_CC\_ATP-SwRS-0721]

EnableDoorOpening\_B，结合远端ATP结果的开门授权信息。

def [EnableDoorOpening\_B](#EnableDoorOpening_B)(k):

return ([LocalATPenableDoorOpening\_B](#LocalATPenableDoorOpening_B)(k)

or [OtherATP](#OtherATP)(k).EnableDoorOpening\_B)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0242]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [EnableDoorOpening\_A](#EnableDoorOpening_A) | √ | √ | BOOLEAN |
| [EnableDoorOpening\_B](#EnableDoorOpening_B) | √ | √ | BOOLEAN |
| [LocalATPenableDoorOpening\_A](#LocalATPenableDoorOpening_A) | √ | √ | BOOLEAN |
| [LocalATPenableDoorOpening\_B](#LocalATPenableDoorOpening_B) | √ | √ | BOOLEAN |
| [TrainEmergencyBrakeApplied](#TrainEmergencyBrakeApplied) | √ | √ | BOOLEAN |
| [TrainInterVPEZ\_A](#TrainInterVPEZ_A) | √ | √ | BOOLEAN |
| [TrainInterVPEZ\_B](#TrainInterVPEZ_B) | √ | √ | BOOLEAN |
| [TrainPackingBrakeApplied](#TrainPackingBrakeApplied) | √ | √ | BOOLEAN |

## F62-Control PSD Opening and Closing Order

由ATP根据CCNV的请求，判断当前是否与指定联锁建立或结束通信。在通信建立成功后，ATP应当将来自CCNV的PSD控制命令由安全通信协议发送给联锁。ATP软件最多同时与联锁建立**MAX\_CONNECTED\_PSD\_NB**个通信连接。

According to the intersection condition between train location and PSD area, ATP shall judge to establish or end the communication with the specified interlock devices. After establishment succeeded, ATP should send the PSD order come from CCNV to the interlock through safe communication protocols. ATP can conduct the **MAX\_CONNECTED\_PSD\_NB** with interlock at the same time.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATOcontrolTimeValid](#ATOcontrolTimeValid) | Internal | F11-Acquire Configuration Data |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [NonVitalRequest](#NonVitalRequest) | External | Interface with CC Non Vital |
| [OtherATP](#OtherATP) | Internal | F13-Manage Loop Hour with Distant ATP |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainFrontLocation](#TrainFrontLocation) | Internal | F33-Confirm Train Localization |
| [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath) | Internal | F33-Confirm Train Localization |
| [TrainRearLocation](#TrainRearLocation) | Internal | F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [PSDmanagerOrder\_A](#PSDmanagerOrder_A) | √ | √ | ST\_PSD\_MANAGE |
| [PSDmanagerOrder\_B](#PSDmanagerOrder_B) | √ | √ | ST\_PSD\_MANAGE |
| [PSDoperation\_A](#PSDoperation_A) | √ | √ | ST\_PSD\_OPERATION |
| [PSDoperation\_B](#PSDoperation_B) | √ | √ | ST\_PSD\_OPERATION |
| [PSDzoneStatus\_A](#PSDzoneStatus_A) | √ | √ | ST\_PSD\_STATUS |
| [PSDzoneStatus\_B](#PSDzoneStatus_B) | √ | √ | ST\_PSD\_STATUS |
| [TableOfPSDPlatform](#TableOfPSDPlatform) | √ | × | ST\_PSD\_TABLE |
| [UsingPSDstatusFromCI](#UsingPSDstatusFromCI) | √ | √ | BOOLEAN |

### Processing

#### Communicate with PSD

判断列车是否与PSD区域有交集，并根据CCNV的请求与PSD通信。

[iTC\_CC\_ATP-SwRS-0266]

AlignPSDzone\_A，列车定位与A侧PSD区有交集；

PSDid\_A，与列车定位有交集的A侧PSD的id号。

* ATP初始化或者失位时，默认设置[AlignPSDzone\_A](#AlignPSDzone_A)为**False**，[PSDid\_A](#PSDid_A)为**None**；
* 当车身定位（即从车尾最小定位到车头最大定位）与屏蔽门区域（线路地图[TrackMap](#TrackMap)中由一对方向相反的**SGL\_PSD\_ZONE**奇点组成）有交集，且根据Table 5‑12判断为A侧时，设置[AlignPSDzone\_A](#AlignPSDzone_A)为**True**，并将[PSDid\_A](#PSDid_A)为设置为该**SGL\_PSD\_ZONE**奇点的id。
* 其他情况，设置[AlignPSDzone\_A](#AlignPSDzone_A)为**False**，[PSDid\_A](#PSDid_A)为**None**。

[AlignPSDzone\_A](#AlignPSDzone_A), ATP shall determine whether there are intersection between the range of train locations and the platform screen doors zone of the side-A.

[PSDid\_A](#PSDid_A), the id of the PSD on side-A intersects with train location.

* In initialization or train delocalization, set [AlignPSDzone\_A](#AlignPSDzone_A) as **False** and [PSDid\_A](#PSDid_A) as **None**.
* If there are intersection between the range of train locations (from the minimum train tail to the maximum train head) and the PSD zone (composing by a pair of **SGL\_PSD\_ZONE** singularities with opposite direction in the train map), and the train door side A correspond to the platform according with Table 5‑12, set [AlignPSDzone\_A](#AlignPSDzone_A) as **True**, and records [PSDid\_A](#PSDid_A) as the PSD’s id:
* Otherwise, set [AlignPSDzone\_A](#AlignPSDzone_A) as **False** and [PSDid\_A](#PSDid_A) as **None**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0261], [iTC\_CC\_ATP\_SwHA-0107], [iTC\_CC\_ATP\_SwHA-0109]

[End]

[iTC\_CC\_ATP-SwRS-0268]

AlignPSDzone\_B，列车定位与B侧PSD区有交集；

PSDid\_B，与列车定位有交集的B侧PSD的id号。

* ATP初始化或者失位时，默认设置[AlignPSDzone\_B](#AlignPSDzone_B)为**False**，[PSDid\_B](#PSDid_B)为**None**；
* 当车身定位（即从车尾最小定位到车头最大定位）与屏蔽门区域（线路地图[TrackMap](#TrackMap)中由一对方向相反的**SGL\_PSD\_ZONE**奇点组成）有交集，且根据Table 5‑12判断为B侧时，设置[AlignPSDzone\_B](#AlignPSDzone_B)为**True**，并将[PSDid\_B](#PSDid_B)为设置为该**SGL\_PSD\_ZONE**奇点的id。
* 其他情况，设置[AlignPSDzone\_B](#AlignPSDzone_B)为**False**，[PSDid\_B](#PSDid_B)为**None**。

[AlignPSDzone\_B](#AlignPSDzone_B), ATP shall determine whether there are intersection between the range of train locations and the platform screen doors zone of the side-A.

[PSDid\_B](#PSDid_B), the id of the PSD on side-A intersects with train location.

* In initialization or train delocalization, set [AlignPSDzone\_B](#AlignPSDzone_B) as **False** and [PSDid\_B](#PSDid_B) as **None**.
* If there are intersection between the range of train locations (from the minimum train tail to the maximum train head) and the PSD zone (composing by a pair of **SGL\_PSD\_ZONE** singularities with opposite direction in the train map), and the train door side B correspond to the platform according with Table 5‑12, set [AlignPSDzone\_B](#AlignPSDzone_B) as **True**, and records [PSDid\_B](#PSDid_B) as the PSD’s id:
* Otherwise, set [AlignPSDzone\_B](#AlignPSDzone_B) as **False** and [PSDid\_B](#PSDid_B) as **None**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0261], [iTC\_CC\_ATP\_SwHA-0107], [iTC\_CC\_ATP\_SwHA-0109]

[End]

[iTC\_CC\_ATP-SwRS-0136]

PSDoperation\_A和PSDoperation\_B，其结构为ST\_PSD\_OPERATION，用于获取来自CCNV的屏蔽门控制指令。

[PSDoperation\_A](#PSDoperation_A) and [PSDoperation\_B](#PSDoperation_B) structured as ST\_PSD\_OPERATION, used to obtain the PSD controlling order from CCNV.

if ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k) == **True**)

[PSDoperation\_A](#PSDoperation_A)(k)= [NonVitalRequest](#NonVitalRequest).[PSDoperation\_A](#PSDoperation_A)(k)

[PSDoperation\_B](#PSDoperation_B)(k)= [NonVitalRequest](#NonVitalRequest).[PSDoperation\_B](#PSDoperation_B)(k)

else:

[PSDoperation\_A](#PSDoperation_A)(k).Id = **None**

[PSDoperation\_B](#PSDoperation_B)(k).Id = **None**

#Category= Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0068], [iTC\_CC-SyAD-1044]

[End]

[iTC\_CC\_ATP-SwRS-0467]

CommunicateWithPSD，ATP根据CCNV的请求，判断是否与联锁建立通信。

* 当本周期来自CCNV的[PSDoperation\_A](#PSDoperation_A)或[PSDoperation\_B](#PSDoperation_B)不全为**None**时，设置CommunicateWithPSD为**True**；
* 否则，设置CommunicateWithPSD为**False**。

ATP shall determine whether to establish communication with the correlative CI according to request from CCNV:

* When there is at least one id of PSDoperation\_A or PSDoperation\_B is not none, ATP shall set CommunicatedWithPSD to **True**:
* Otherwise, set CommunicatedWithPSD to **False**.

def [CommunicateWithPSD](#CommunicateWithPSD)(k):

return ([PSDoperation\_A](#PSDoperation_A)(k).Id is not **None**

or [PSDoperation\_B](#PSDoperation_B)(k).Id is not **None**)

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0068], [iTC\_CC-SyAD-0265]

[End]

当列车运行在配置有PSD的车站时，ATP需与联锁建立通信，获取PSD的状态信息。根据配置数据，ATP软件维护来自CI的PSD的状态信息数组TableOfPSDPlatform，其索引就是PSD的id值，每个数组元素如ST\_PSD\_TABLE所示：

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_PSD\_TABLE | |  |  |
|  | DoorStatusValidityTime | NUMERIC\_32 | PSD状态是否有效 |
|  | DoorClosed | BOOLEAN | PSD状态 |

When the train is running in the station equipped with PSD, ATP needs to communicate with CI and get the PSD status information. According to the configuration data, ATP will process the PSD from CI, and the index is the id of PSD, just as shown in the ST\_PSD\_TABLE

[iTC\_CC\_ATP-SwRS-0111]

初始化时，设置[TableOfPSDPlatform](#TableOfPSDPlatform)数组中所有PSD的DoorClosed均为**False**，其有效期为0；此后，如果本周期收到正确的来自CI的[CI\_IOstatus](#CI_IOstatus)消息时，根据其ID号，更新[TableOfPSDPlatform](#TableOfPSDPlatform)数组中相应PSD的DoorClosed状态，并将DoorStatusValidityTime设为[ATPsetting](#ATPsetting).PSDstatusValidityTime减去FSFB2消息传输延迟。

In initialization, all PSD DoorClosed of [TableOfPSDPlatform](#TableOfPSDPlatform) set as **False**, and the valid period is zero. Later on, if receiving correct [CI\_IOstatus](#CI_IOstatus), on the basis of ID number the status of related PSD doorClosed from [TableOfPSDPlatform](#TableOfPSDPlatform) is updated and the DoorStatusValidityTime is set as [ATPsetting](#ATPsetting).PSDstatusValidityTime subtracted the delay of FSFB2 message.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0949], [iTC\_CC\_ATP\_SwHA-0032], [iTC\_CC\_ATP\_SwHA-0276]

[End]

[iTC\_CC\_ATP-SwRS-0112]

如果本周期未收到来自CI的[CI\_IOstatus](#CI_IOstatus)消息，或者该消息校验错误，则ATP更新[TableOfPSDPlatform](#TableOfPSDPlatform)数组中的PSD的状态。即将DoorStatusValidityTime减1，如果该值已小于等于0，则将DoorClosed 设为**False**；否则保持DoorClosed不变。

If ATP does not receive the [CI\_IOstatus](#CI_IOstatus) from CI, or if this message detected as **False**, ATP shall update the PSD status of [TableOfPSDPlatform](#TableOfPSDPlatform), i.e. it is necessary to minus DoorStatusValidityTime with one. If the value is less than or equal to zero, the status of DoorClosed is set as **False**; otherwise the status keeps the same.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0032], [iTC\_CC\_ATP\_SwHA-0276]

[End]

#### PSD status

在CBTC模式下，当列车在运动过程中，ATP均相信并采用来自ZC的PSD变量状态作为该PSD的状态信息；而在列车静止或刚发车时，应采用与联锁通信得到的PSD结果。在BLOCK模式下，在列车运行时，ATP不考虑该PSD的状态（即认为该PSD是关闭并锁闭的）；在停稳或刚发车时，再使用来自联锁的PSD状态信息。因此，本功能更新的PSDzoneStatus\_A和PSDzoneStatus\_B信息，需根据本端ATP是否主机，以及来自联锁、ZC或是BM信标的信息判断PSD状态。PSD状态结构体为ST\_PSD\_STATUS：

In CBTC mode, when train is moving, ATP shall monitor the PSD status by the variant from [ZC](#ZC); and on the other hand, when train is stopping or just beginning to start, ATP shall use the PSD status adopted from the communication with interlock device. In the block mode, ATP will not consider the PSD status when train is moving (i.e. Will consider the PSD is closed and locked); and when train is stopping or starting, ATP will use the PSD information from interlock devices. Therefore, ATP should consider the updating of [PSDzoneStatus\_A](#PSDzoneStatus_A) and [PSDzoneStatus\_B](#PSDzoneStatus_B) based on the whether ATP is master or not, the information from interlock device, [ZC](#ZC) or BM.

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_PSD\_STATUS | |  |  |
|  | Id | NUMERIC\_32 | PSD的标识 |
|  | Validity | NUMERIC\_32 | PSD剩余有效期 |
|  | AllPSDclosed | BOOLEAN | PSD状态 |

[iTC\_CC\_ATP-SwRS-0332]

UsingPSDstatusFromCI，只有当列车定位与PSD区域有交集，且列车静止或刚发车时，ATP使用来自CI的PSD状态信息。

Only when the train fulfilled the following conditions, ATP shall use the PSD status from the CI:

* The train location intersects with a PSD zone;
* And the train is filtered stopped or just started moving.

[UsingPSDstatusFromCI](#UsingPSDstatusFromCI) = (([AlignPSDzone\_A](#AlignPSDzone_A)(k) or [AlignPSDzone\_B](#AlignPSDzone_B)(k))

and ([TrainFilteredStopped](#TrainFilteredStopped)(k)

or ([TrainFilteredStopped](#TrainFilteredStopped)(k-1)

and not [TrainFilteredStopped](#TrainFilteredStopped)(k))))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0266]

[End]

[iTC\_CC\_ATP-SwRS-0134]

MasterCCcore，来自CCNV的当前是否为主控CC信息

[MasterCCcore](#MasterCCcore) shows whether the status from CCNV is the main controlled CC.

if ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k) == **True**)

[MasterCCcore](#MasterCCcore) = [NonVitalRequest](#NonVitalRequest).MasterCcCore(k)

else:

[MasterCCcore](#MasterCCcore) = **False**

#Category= Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0068], [iTC\_CC-SyAD-1044]

[End]

[iTC\_CC\_ATP-SwRS-0155]

[PSDzoneStatus\_A](#PSDzoneStatus_A), 如果ATP所在为主控CC，则对A侧PSD状态的更新规则如下：

If the ATP is the master CC, then the A-side PSD state updating rules are as follows:

if ([MasterCCcore](#MasterCCcore)(k) == **True**)

if (([PSDoperation\_A](#PSDoperation_A).Id == **None**) or ([PSDoperation\_A](#PSDoperation_A).Id != [PSDid\_A](#PSDid_A)(k)))

[PSDzoneStatus\_A](#PSDzoneStatus_A).Id(k)= **None**

[PSDzoneStatus\_A](#PSDzoneStatus_A).Validity = **0**

[PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed = **False**

else:

[PSDzoneStatus\_A](#PSDzoneStatus_A)(k).Id = [PSDoperation\_A](#PSDoperation_A)(k).Id

if ([UsingPSDstatusFromCI](#UsingPSDstatusFromCI)(k))

[PSDzoneStatus\_A](#PSDzoneStatus_A).Validity(k) = [TableOfPSDPlatform](#TableOfPSDPlatform)[[PSDid\_A](#PSDid_A)(k)].DoorStatusValidityTime

[PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed(k) = [TableOfPSDPlatform](#TableOfPSDPlatform)[[PSDid\_A](#PSDid_A)(k)].DoorClosed

elif ([CoercedPermissive](#CoercedPermissive)([TrackMap](#TrackMap).PSDs[[PSDid\_A](#PSDid_A)(k)].CoercedPermissive, k))

[PSDzoneStatus\_A](#PSDzoneStatus_A).Validity = **REPORT\_AGE\_MAX**

[PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed = **True**

elif ([CoercedRestrictive](#CoercedRestrictive)([TrackMap](#TrackMap).PSDs[[PSDid\_A](#PSDid_A)(k)].NotCoercedRestrictive, k))

[PSDzoneStatus\_A](#PSDzoneStatus_A).Validity = **REPORT\_AGE\_MAX**

[PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed = **False**

else:

[PSDzoneStatus\_A](#PSDzoneStatus_A).Validity(k) = [ReceivedVariantReport](#ReceivedVariantReport)[LineSectionOfPSD].ValidityTime

[PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed(k)= [VariantValue](#VariantValue)([TrackMap](#TrackMap).PSDs[[PSDid\_A](#PSDid_A)(k)].Variant, k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0137], [iTC\_CC\_ATP\_SwHA-0138]

[End]

[iTC\_CC\_ATP-SwRS-0532]

[PSDzoneStatus\_B](#PSDzoneStatus_B), 如果ATP所在为主控CC，则对B侧PSD状态的更新规则如下：

If the ATP is the master CC, then the B-side PSD state updating rules are as follows:

if ([MasterCCcore](#MasterCCcore)(k) == **True**)

if (([PSDoperation\_B](#PSDoperation_B).Id == **None**) or ([PSDoperation\_B](#PSDoperation_B).Id != [PSDid\_B](#PSDid_B)(k)))

[PSDzoneStatus\_B](#PSDzoneStatus_B).Id(k)= **None**

[PSDzoneStatus\_B](#PSDzoneStatus_B).Validity = 0

[PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed = **False**

else:

[PSDzoneStatus\_B](#PSDzoneStatus_B).Id(k)= [PSDoperation\_B](#PSDoperation_B).Id

if ([UsingPSDstatusFromCI](#UsingPSDstatusFromCI)(k))

[PSDzoneStatus\_B](#PSDzoneStatus_B).Validity(k) = [TableOfPSDPlatform](#TableOfPSDPlatform)[[PSDid\_B](#PSDid_B)(k)].DoorStatusValidityTime

[PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed(k) = [TableOfPSDPlatform](#TableOfPSDPlatform)[[PSDid\_B](#PSDid_B)(k)].DoorClosed

elif ([CoercedPermissive](#CoercedPermissive)([TrackMap](#TrackMap).PSDs[[PSDid\_B](#PSDid_B)(k)].CoercedPermissive, k))

[PSDzoneStatus\_B](#PSDzoneStatus_B).Validity = **REPORT\_AGE\_MAX**

[PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed = **True**

elif ([CoercedRestrictive](#CoercedRestrictive)([TrackMap](#TrackMap).PSDs[[PSDid\_B](#PSDid_B)(k)].NotCoercedRestrictive, k))

[PSDzoneStatus\_B](#PSDzoneStatus_B).Validity =  **REPORT\_AGE\_MAX**

[PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed = **False**

else:

[PSDzoneStatus\_B](#PSDzoneStatus_B).Validity(k) = [ReceivedVariantReport](#ReceivedVariantReport)[LineSectionOfPSD].ValidityTime

[PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed(k)= [VariantValue](#VariantValue)([TrackMap](#TrackMap).PSDs[[PSDid\_B](#PSDid_B)(k)].Variant, k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0137], [iTC\_CC\_ATP\_SwHA-0138]

[End]

[iTC\_CC\_ATP-SwRS-0533]

如果ATP所在为备机CC，则对A侧PSD状态的更新规则如下：

If the ATP is not the master CC, then the A-side PSD state updating rules are as follows:

if ([MasterCCcore](#MasterCCcore) != **True**)

if ([OtherATPmessageAvailable](#OtherATPmessageAvailable)(k) == **True**)

[PSDzoneStatus\_A](#PSDzoneStatus_A).Id = [OtherATP](#OtherATP).PsdIdSide\_A

if ([PSDzoneStatus\_A](#PSDzoneStatus_A).Id(k) != **None**)

[PSDzoneStatus\_A](#PSDzoneStatus_A).Validity(k) = ([OtherATP](#OtherATP).PsdValiditySide\_A

- [Message.ModularSub](#ModularSub)([ATPtime](#ATPtime)(k), [OtherATP](#OtherATP).LatestTimeOtherCore))

[PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed(k) = ([OtherATP](#OtherATP).PsdClosedSide\_A

and ([PSDzoneStatus\_A](#PSDzoneStatus_A).Validity(k) > 0))

else:

[PSDzoneStatus\_A](#PSDzoneStatus_A).Validity = 0

[PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed = **False**

elif ([PSDzoneStatus\_A](#PSDzoneStatus_A).Id(k-1) != **None**)

[PSDzoneStatus\_A](#PSDzoneStatus_A).Id = [PSDzoneStatus\_A](#PSDzoneStatus_A).Id(k-1)

[PSDzoneStatus\_A](#PSDzoneStatus_A).Validity = [PSDzoneStatus\_A](#PSDzoneStatus_A).Validity(k-1)- 1

[PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed(k) = ([PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed(k-1)

and ([PSDzoneStatus\_A](#PSDzoneStatus_A).Validity(k) > 0))

else:

[PSDzoneStatus\_A](#PSDzoneStatus_A).Validity = 0

[PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC\_ATP\_SwHA-0138]

[End]

[iTC\_CC\_ATP-SwRS-0534]

如果ATP所在为备机CC，则对B侧PSD状态的更新规则如下：

If the ATP is not the master CC, then the B-side PSD state updating rules are as follows:

if ([MasterCCcore](#MasterCCcore) != **True**)

if ([OtherATPmessageAvailable](#OtherATPmessageAvailable)(k) == **True**)

[PSDzoneStatus\_B](#PSDzoneStatus_B).Id = [OtherATP](#OtherATP).PsdIdSide\_B

if ([PSDzoneStatus\_B](#PSDzoneStatus_B).Id(k) != **None**)

[PSDzoneStatus\_B](#PSDzoneStatus_B).Validity(k) = ([OtherATP](#OtherATP).PsdValiditySide\_B

- [Message.ModularSub](#ModularSub)([ATPtime](#ATPtime)(k), [OtherATP](#OtherATP).LatestTimeOtherCore))

[PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed(k)

= [OtherATP](#OtherATP).PsdClosedSide\_B

and ([PSDzoneStatus\_B](#PSDzoneStatus_B).Validity(k) > 0)

else:

[PSDzoneStatus\_B](#PSDzoneStatus_B).Validity = 0

[PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed = **False**

elif ([PSDzoneStatus\_B](#PSDzoneStatus_B).Id(k-1) != **None**)

[PSDzoneStatus\_B](#PSDzoneStatus_B).Id = [PSDzoneStatus\_B](#PSDzoneStatus_B).Id(k-1)

[PSDzoneStatus\_B](#PSDzoneStatus_B).Validity = [PSDzoneStatus\_B](#PSDzoneStatus_B).Validity(k-1) - 1

[PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed(k)

= [PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed(k-1)

and ([PSDzoneStatus\_B](#PSDzoneStatus_B).Validity(k) > 0)

else:

[PSDzoneStatus\_B](#PSDzoneStatus_B).Validity = 0

[PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0138]

[End]

[iTC\_CC\_ATP-SwRS-0722]

PSDstatusNonVital\_A，用于CCNV发送给DMI显示的A侧PSD状态

def [PSDstatusNonVital\_A](#PSDstatusNonVital_A)(k):

if ([PSDzoneStatus\_A](#PSDzoneStatus_A)(k).Id is **None**

or [PSDzoneStatus\_A](#PSDzoneStatus_A)(k).Validity <= 0):

return **PSD\_STATE\_UNKNOWN**

elif ([PSDzoneStatus\_A](#PSDzoneStatus_A)(k).AllPSDclosed):

return **PSD\_STATE\_CLOSED**

else:

return **PSD\_STATE\_OPENED**

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0408], [iTC\_CC-SyAD-0267]

[End]

[iTC\_CC\_ATP-SwRS-0723]

PSDstatusNonVital\_B，用于CCNV发送给DMI显示的B侧PSD状态

def [PSDstatusNonVital\_B](#PSDstatusNonVital_B)(k):

if ([PSDzoneStatus\_B](#PSDzoneStatus_B)(k).Id is **None**

or [PSDzoneStatus\_B](#PSDzoneStatus_B)(k).Validity <= 0):

return **PSD\_STATE\_UNKNOWN**

elif ([PSDzoneStatus\_B](#PSDzoneStatus_B)(k).AllPSDclosed):

return **PSD\_STATE\_CLOSED**

else:

return **PSD\_STATE\_OPENED**

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0408], [iTC\_CC-SyAD-0267]

[End]

#### PSD order

对于PSD的开关门指令，ATP根据来自CCNV的指令，根据项目配置的安全通信协议进行发送。

For the control order of the PSD, ATP shall generate according to the request of the CCNV and the safety communication protocol defined in the project configuration.

[iTC\_CC\_ATP-SwRS-0333]

PSDmanagerOrder\_A，A侧PSD的控制命令信息，其结构为ST\_PSD\_MANAGE。其中如果来自CCNV的A侧PSD标识不等于ATP读取SGD中A侧的标识，则禁止使用CCNV的标识开门。

The rules to generate the PSD manage order on side A shall follow the pseudo-codes. In which if the PSD id from CCNV is not equal to the id in ATP's track map, ATP shall prohibit the PSD opening.

def [PSDmanagerOrder\_A](#PSDmanagerOrder_A)(k):

[PSDmanagerOrder\_A](#PSDmanagerOrder_A).Id = [PSDoperation\_A](#PSDoperation_A).Id(k)

if ([PSDoperation\_A](#PSDoperation_A).Id(k) == [PSDid\_A](#PSDid_A)(k)

and [PSDoperation\_A](#PSDoperation_A).Id(k) is not **None**

and not [PSDoperation\_A](#PSDoperation_A).ClosingOrder(k)

and [PSDoperation\_A](#PSDoperation_A).OpeningOrder(k)

and [EnableDoorOpening\_A](#EnableDoorOpening_A)(k)):

[PSDmanagerOrder\_A](#PSDmanagerOrder_A).Order = Open\_PSD\_Configuration

elif (not [PSDoperation\_A](#PSDoperation_A).OpeningOrder(k)

and [PSDoperation\_A](#PSDoperation_A).ClosingOrder(k)):

[PSDmanagerOrder\_A](#PSDmanagerOrder_A).Order = Close\_PSD\_Of\_Platform

else:

[PSDmanagerOrder\_A](#PSDmanagerOrder_A).Order = **None**

return [PSDmanagerOrder\_A](#PSDmanagerOrder_A)

In the above ARDL, the Open\_PSD\_Configuration and Close\_PSD\_Of\_Platform are control words defined in the project configuration.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0265], [iTC\_CC\_ATP\_SwHA-0139]

[End]

[iTC\_CC\_ATP-SwRS-0334]

PSDmanagerOrder\_B，B侧PSD的控制命令信息，其结构为ST\_PSD\_MANAGE。其中如果来自CCNV的B侧PSD标识不等于ATP读取SGD中B侧的标识，则禁止使用来自CCNV的标识开门。

The rules to generate the PSD manage order on side B shall follow the pseudo-codes. In which if the PSD id from CCNV is not equal to the id in ATP's track map, ATP shall prohibit the PSD opening.

def [PSDmanagerOrder\_B](#PSDmanagerOrder_B)(k):

[PSDmanagerOrder\_B](#PSDmanagerOrder_B).Id = [PSDoperation\_B](#PSDoperation_B).Id(k)

if ([PSDoperation\_B](#PSDoperation_B).Id(k) == [PSDid\_B](#PSDid_B)(k)

and [PSDoperation\_B](#PSDoperation_B).Id(k) is not **None**

and not [PSDoperation\_B](#PSDoperation_B).ClosingOrder(k)

and [PSDoperation\_B](#PSDoperation_B).OpeningOrder(k)

and [EnableDoorOpening\_B](#EnableDoorOpening_B)(k)):

[PSDmanagerOrder\_B](#PSDmanagerOrder_B).Order = Open\_PSD\_Configuration

elif (not [PSDoperation\_B](#PSDoperation_B).OpeningOrder(k)

and [PSDoperation\_B](#PSDoperation_B).ClosingOrder(k)):

[PSDmanagerOrder\_B](#PSDmanagerOrder_B).Order = Close\_PSD\_Of\_Platform

else:

[PSDmanagerOrder\_B](#PSDmanagerOrder_B).Order = **None**

return [PSDmanagerOrder\_B](#PSDmanagerOrder_B)

In the above ARDL, the Open\_PSD\_Configuration and Close\_PSD\_Of\_Platform are control words defined in the project configuration.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0265], [iTC\_CC\_ATP\_SwHA-0139]

[End]

[iTC\_CC\_ATP-SwRS-0335]

PSDplatformManagerOpeningOrder，本ATP是否发了开门命令.

ATP shall determine whether itself opening the PSD in this cycle.

if (Initialization)

[PSDplatformManagerOpeningOrder](#PSDplatformManagerOpeningOrder) = **False**

elif (([PSDmanagerOrder\_A](#PSDmanagerOrder_A).Order(k) == Open\_PSD\_Configuration)

or ([PSDmanagerOrder\_B](#PSDmanagerOrder_B).Order(k) == Open\_PSD\_Configuration))

[PSDplatformManagerOpeningOrder](#PSDplatformManagerOpeningOrder) = **True**

else:

[PSDplatformManagerOpeningOrder](#PSDplatformManagerOpeningOrder) = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0230]

[End]

#### Send PSD command

[iTC\_CC\_ATP-SwRS-0336]

PSDopeningCommand，本ATP或者冗余端ATP当前是否在发送开PSD命令.

ATP shall determine whether itself or the redundant ATP opening the PSD in this cycle.

if (([PSDplatformManagerOpeningOrder](#PSDplatformManagerOpeningOrder)(k) == **True**)

or ([OtherATP](#OtherATP).PsdManagerOpeningOrder(k) == **True**))

[PSDopeningCommand](#PSDopeningCommand) = **True**

else:

[PSDopeningCommand](#PSDopeningCommand) = **False**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0230]

[End]

[iTC\_CC\_ATP-SwRS-0444]

在与联锁通信时，如果[PSDmanagerOrder\_A](#PSDmanagerOrder_A).Id有效，则根据[PSDmanagerOrder\_A](#PSDmanagerOrder_A).Order和离线配置数据设置发送给A侧屏蔽门的控制信息[CIsetting](#CIsetting)[0]。

When communicating with the CI, if the [PSDmanagerOrder\_A](#PSDmanagerOrder_A) which comes from CCNV was valid, ATP shall set the [CIsetting](#CIsetting)[0] according to the [PSDmanagerOrder\_A](#PSDmanagerOrder_A) and the configuration of the PSD.

if (([CommunicateWithPSD](#CommunicateWithPSD)(k) == **True**)

and ([PSDmanagerOrder\_A](#PSDmanagerOrder_A).Id != **None**))

[CIsetting](#CIsetting)[0].PlatformId = [PSDmanagerOrder\_A](#PSDmanagerOrder_A).Id

if ([PSDmanagerOrder\_A](#PSDmanagerOrder_A).Order == Open\_PSD\_Configuration)

[CIsetting](#CIsetting)[0].Order = [TrackMap](#TrackMap).PSDs[[PSDmanagerOrder\_A](#PSDmanagerOrder_A).Id].DoorOpeningCode

elif ([PSDmanagerOrder\_A](#PSDmanagerOrder_A).Order == Close\_PSD\_Configuration):

[CIsetting](#CIsetting)[0].Order = [TrackMap](#TrackMap).PSDs[[PSDmanagerOrder\_A](#PSDmanagerOrder_A).Id].DoorClosingCode

else:

[CIsetting](#CIsetting)[0].Order = [TrackMap](#TrackMap).PSDs[[PSDmanagerOrder\_A](#PSDmanagerOrder_A).Id].DoorNoActionCode

else:

[CIsetting](#CIsetting)[0].PlatformId = **None**

[CIsetting](#CIsetting)[0].Order = **None**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0265]

[End]

[iTC\_CC\_ATP-SwRS-0445]

在与联锁通信时，如果[PSDmanagerOrder\_B](#PSDmanagerOrder_B).Id有效，则根据[PSDmanagerOrder\_B](#PSDmanagerOrder_B).Order和离线配置数据设置发送给B侧屏蔽门的控制信息[CIsetting](#CIsetting) [1]。

When communicating with the CI, if the [PSDmanagerOrder\_B](#PSDmanagerOrder_B), which comes from CCNV, was valid, ATP shall set the [CIsetting](#CIsetting)[1] according to the [PSDmanagerOrder\_B](#PSDmanagerOrder_B) and the configuration of the PSD.

if (([CommunicateWithPSD](#CommunicateWithPSD)(k) == **True**)

and ([PSDmanagerOrder\_B](#PSDmanagerOrder_B).Id != **None**)):

[CIsetting](#CIsetting)[1].PlatformId = [PSDmanagerOrder\_B](#PSDmanagerOrder_B).Id

if ([PSDmanagerOrder\_B](#PSDmanagerOrder_B).Order == Open\_PSD\_Configuration)

[CIsetting](#CIsetting)[1].Order = [TrackMap](#TrackMap).PSDs[[PSDmanagerOrder\_B](#PSDmanagerOrder_B).Id].DoorOpeningCode

elif ([PSDmanagerOrder\_B](#PSDmanagerOrder_B).Order == Close\_PSD\_Configuration):

[CIsetting](#CIsetting)[1].Order = [TrackMap](#TrackMap).PSDs[[PSDmanagerOrder\_B](#PSDmanagerOrder_B).Id].DoorClosingCode

else:

[CIsetting](#CIsetting)[1].Order = [TrackMap](#TrackMap).PSDs[[PSDmanagerOrder\_B](#PSDmanagerOrder_B).Id].DoorNoActionCode

else:

[CIsetting](#CIsetting)[1].PlatformId = **None**

[CIsetting](#CIsetting)[1].Order = **None**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0265]

[End]

### Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Safety Critical | | Observable | | | Logical Type |
| [AlignPSDzone\_A](#AlignPSDzone_A) | | √ | | √ | BOOLEAN | |
| [AlignPSDzone\_B](#AlignPSDzone_B) | | √ | | √ | BOOLEAN | |
| [CommunicateWithPSD](#CommunicateWithPSD) | | √ | | √ | BOOLEAN | |
| [MasterCCcore](#MasterCCcore) | | × | | √ | BOOLEAN | |
| [PSDid\_A](#PSDid_A) | | √ | | √ | NUMERIC\_32 | |
| [PSDid\_B](#PSDid_B) | | √ | | √ | NUMERIC\_32 | |
| [PSDopeningCommand](#PSDopeningCommand) | | √ | | √ | BOOLEAN | |
| [PSDplatformManagerOpeningOrder](#PSDplatformManagerOpeningOrder) | | √ | | √ | BOOLEAN | |
| [PSDstatusNonVital\_A](#PSDstatusNonVital_A) | | × | | √ | ENUM\_PSD\_STATE | |
| [PSDstatusNonVital\_B](#PSDstatusNonVital_B) | | × | | √ | ENUM\_PSD\_STATE | |

## F63-Monitor Status of Doors

本模块用于描述ATP软件监控车门和屏蔽门开启的需求。

This module describes the ATP requirements for the monitor status of doors.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [AlignPSDzone\_A](#AlignPSDzone_A) | Internal | F62-Control PSD Opening and Closing Order |
| [AlignPSDzone\_B](#AlignPSDzone_B) | Internal | F62-Control PSD Opening and Closing Order |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainInterVPEZ\_A](#TrainInterVPEZ_A) | Internal | F61-Elaborate Door Opening Authorization |
| [TrainInterVPEZ\_B](#TrainInterVPEZ_B) | Internal | F61-Elaborate Door Opening Authorization |
| [TrainLocalized](#TrainLocalized) | Internal | F32-Update Train Location |
| [TrainParkingBrakeApplied](#TrainParkingBrakeApplied) | Internal | F61-Elaborate Door Opening Authorization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [AllPSDclosedAndLocked](#AllPSDclosedAndLocked) | √ | √ | BOOLEAN |
| [AllTrainDoorsClosedAndLocked](#AllTrainDoorsClosedAndLocked) | √ | √ | BOOLEAN |
| [PSDopeningCommandValid](#PSDopeningCommandValid) | √ | √ | BOOLEAN |

### Processing

#### Train doors monitoring

若列车停在车站，当车门开启时，ATP应当保持输出停车制动

When the train stopped and opened the door in the station, ATP shall keep outputting parking brake.

[iTC\_CC\_ATP-SwRS-0070]

AllTrainDoorsClosedAndLocked，两端车头有任意一端采到TDCL，即认为两侧车门关闭并锁闭。

The [AllTrainDoorsClosedAndLocked](#AllTrainDoorsClosedAndLocked) stands for the condition that either side of both train ends collect TDCL, i.e. both side of door is closed and locked.

def [AllTrainDoorsClosedAndLocked](#AllTrainDoorsClosedAndLocked)(k):

return [Offline.GetAllTrainDoorsClosedAndLocked](#GetAllTrainDoorsClosedAndLocked)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0239], [iTC\_CC-SyAD-0241], [iTC\_CC-SyAD-1003], [iTC\_CC\_ATP\_SwHA-0268]

[End]

[iTC\_CC\_ATP-SwRS-0337]

InhibitControlTrainDoorsStatus，不监控车门状态.

ATP shall not monitor the status of train doors when InhibitControlTrainDoorsStatus is selected.

def [InhibitControlTrainDoorsStatus](#InhibitControlTrainDoorsStatus)(k):

return [Offline.GetInhibitControlTrainDoorsStatus](#GetInhibitControlTrainDoorsStatus)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1220], [iTC\_CC-SyAD-1303]

[End]

[iTC\_CC\_ATP-SwRS-0804]

NoDangerForTrainDoorsNotClosedAndLocked，当列车停车，且与PSD区或VPEZ有交集时，且TDCL丢失，则该值为假；否则，该值为真。

def [NoDangerForTrainDoorsNotClosedAndLocked](#NoDangerForTrainDoorsNotClosedAndLocked)(k):

return not ([TrainFilteredStopped](#TrainFilteredStopped)(k)

and not [AllTrainDoorsClosedAndLocked](#AllTrainDoorsClosedAndLocked)(k)

and ([AlignPSDzone\_A](#AlignPSDzone_A)(k) or [AlignPSDzone\_B](#AlignPSDzone_B)(k)

or [TrainInterVPEZ\_A](#TrainInterVPEZ_A)(k) or [TrainInterVPEZ\_B](#TrainInterVPEZ_B)(k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0230], [iTC\_CC-SyAD-0231]

[End]

[iTC\_CC\_ATP-SwRS-0338]

PBforTrainDoorsNotClosedAndLocked，列车停车，且车身与PSD区或VPEZ区域有交集时，车门未关时保持PB输出。

If the train is aligning in a PSD or intersecting with a vital passage exchange zone, and the RMF or RMR does not selected, ATP shall keep triggering parking brake when the train doors does not closed and locked.

def [PBforTrainDoorsNotClosedAndLocked](#PBforTrainDoorsNotClosedAndLocked)(k):

return (not [NoDangerForTrainDoorsNotClosedAndLocked](#NoDangerForTrainDoorsNotClosedAndLocked)(k)

and not [InhibitControlTrainDoorsStatus](#InhibitControlTrainDoorsStatus)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0230], [iTC\_CC-SyAD-0231], [iTC\_CC-SyAD-1220], [iTC\_CC-SyAD-1226], [iTC\_CC\_ATP\_SwHA-0141]

[End]

[iTC\_CC\_ATP-SwRS-0339]

EBforPBnotAppliedDueToTrainDoors，由于车门开而输出ZVRD，但未检测到ZVBA, 则ATP应当输出EB.

If ATP has triggered parking brake for train doors opening, but it does not applied by the rolling stock, ATP shall trigger the emergency brake.

[EBforPBnotAppliedDueToTrainDoors](#EBforPBnotAppliedDueToTrainDoors)(k)

= (([PBforTrainDoorsNotClosedAndLocked](#PBforTrainDoorsNotClosedAndLocked)(k) == **True**)

and ([TrainParkingBrakeApplied](#TrainParkingBrakeApplied)(k) != **True**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0233], [iTC\_CC\_ATP\_SwHA-0142]

[End]

[iTC\_CC\_ATP-SwRS-0340]

NoDangerForDepartureWithoutTDCL，判断是否未处于上周期停车而本周期开始动车，且车门未关的条件。

ATP shall determine whether the train is departure without TDCL.

def [NoDangerForDepartureWithoutTDCL](#NoDangerForDepartureWithoutTDCL)(k):

return ([AllTrainDoorsClosedAndLocked](#AllTrainDoorsClosedAndLocked)(k)

or [TrainFilteredStopped](#TrainFilteredStopped)(k)

or not [TrainFilteredStopped](#TrainFilteredStopped)(k-1))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0143]

[End]

[iTC\_CC\_ATP-SwRS-0749]

EBforDepartureWithoutTDCL，若ATP监控发车时丢失TDCL的情况，则输出EB。

If ATP needs to monitor the status of train doors, ATP shall trigger EB if train determine without TDCL:

def [EBforDepartureWithoutTDCL](#EBforDepartureWithoutTDCL)(k):

return (not [NoDangerForDepartureWithoutTDCL](#NoDangerForDepartureWithoutTDCL)(k)

and not [InhibitControlTrainDoorsStatus](#InhibitControlTrainDoorsStatus)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1226], [iTC\_CC-SyAD-1227], [iTC\_CC\_ATP\_SwHA-0143]

[End]

**NOTES：**

对于非授权开门状态下车门打开（MWTDCL）监控与[EvacuationWhileLeavingStation](#EvacuationWhileLeavingStation)功能（EWLS）的区别及其配置说明：

* 已有的EWLS功能SwRS-0365，在离站距离内（车门授权DE由真变假后开始记录），车动且乘客手柄丢失，会EB；
* 而MWTDCL，与之有两个区别：
* 车辆输入的乘客紧急手柄[EmergencyHandleNotPulledSide](#EmergencyHandleNotPulledSide)和车门状态[AllTrainDoorsClosedAndLocked](file:///D:\ATP\3-Requirement\ATP-SwRS.docx#AllTrainDoorsClosedAndLocked)是否合一；
* 项目能否实现Door opening authorization开门授权功能（例如不采集ZVBA输入，则不会满足开门授权条件）。
* 因此，可根据项目要求，选择配置MWTDCL和EWLS这两个功能，如下表所示。

Table 5‑13 Configuration for monitoring unexpected train door open

|  |  |  |
| --- | --- | --- |
| Project configuration | With Door opening authorization | Without Door opening authorization |
| Source of EHNPS and TDCL from RS are same | Current metro projects:  -- Can modify evacuation station area length for EWLS to fulfill the project  -- MWTDCL is unnecessary | Tramcar project for Ethiopia:  -- EWLS is disabled  -- Set MWTDCL according to project |
| Source of EHNPS and TDCL from RS are independent | Maybe future metro projects:  -- set EWLS to monitor EHNPS  -- set MWTDCL to monitor TDCL |

[iTC\_CC\_ATP-SwRS-0799]

InhibitProtectionMovingWithoutTDCL，禁止监控非开门授权情况下车门打开的情形。

ATP shall not monitor the status of train doors open without door opening enable if [InhibitProtectionMovingWithoutTDCL](#InhibitProtectionMovingWithoutTDCL) is selected.

def [InhibitProtectionMovingWithoutTDCL](#InhibitProtectionMovingWithoutTDCL)(k):

return Offline.[GetInhibitProtectionMovingWithoutTDCL](#GetInhibitProtectionMovingWithoutTDCL)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1441]

[End]

[iTC\_CC\_ATP-SwRS-0800]

NoDangerForMovingWithoutTDCL，监控非授权开门状态下车门打开

def [NoDangerForMovingWithoutTDCL](#NoDangerForMovingWithoutTDCL)(k):

return ([AllTrainDoorsClosedAndLocked](#AllTrainDoorsClosedAndLocked)(k)

or [TrainFilteredStopped](#TrainFilteredStopped)(k)

or [EnableDoorOpening\_A](#EnableDoorOpening_A)(k)

or [EnableDoorOpening\_B](#EnableDoorOpening_B)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1441]

[End]

[iTC\_CC\_ATP-SwRS-0801]

EBforMovingWithoutTDCL，禁止监控非开门授权情况下车门打开的情形。

def [EBforMovingWithoutTDCL](#EBforMovingWithoutTDCL)(k):

return (not [NoDangerForMovingWithoutTDCL](#NoDangerForMovingWithoutTDCL)(k)

and not [InhibitProtectionMovingWithoutTDCL](#InhibitProtectionMovingWithoutTDCL)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1441]

[End]

#### PSD status monitoring

在非RM模式，或BLOCK模式要求与PSD进行通信的条件下， ATP应当确保在PSD开启，或者发送PSD开启命令时，列车不能发生运动，否则将触发EB。

In the non-RM mode, or in block mode with PSD communication, ATP shall guarantee the train cannot move while the PSD has opened or the opening PSD order is sending. Otherwise, ATP shall trigger EB.

[iTC\_CC\_ATP-SwRS-0341]

InhibitControlPSDstatus，项目可配置不监控PSD状态的条件。

The conditions ATP does not control PSD can be configured by project.

def [InhibitControlPSDstatus](#InhibitControlPSDstatus)(k):

return [Offline.GetInhibitControlPSDstatus](#GetInhibitControlPSDstatus)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1221], [iTC\_CC-SyAD-1230], [iTC\_CC-SyAD-1304], [iTC\_CC\_ATP\_SwHA-0269]

[End]

[iTC\_CC\_ATP-SwRS-0342]

AllPSDclosedAndLocked的判断，上周期或本周期停车，若有PSD且已获取其状态为关闭。

If the train stopped or just started moving, and the status of all aligned PSD are closed, ATP shall consider the [AllPSDclosedAndLocked](#AllPSDclosedAndLocked) is **True**.

[AllPSDclosedAndLocked](#AllPSDclosedAndLocked)(k)

= (([TrainLocalized](#TrainLocalized)(k) == **True**)

and (([TrainFilteredStopped](#TrainFilteredStopped)(k) == **True**)

or ([TrainFilteredStopped](#TrainFilteredStopped)(k-1) == **True**))

and (([PSDid\_A](#PSDid_A)(k) == 0) and ([PSDid\_B](#PSDid_B)(k) == 0))

or (([PSDid\_A](#PSDid_A)(k) != 0) and ([PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed(k) == **True**)

and ([PSDid\_B](#PSDid_B)(k) == 0))

or (([PSDid\_B](#PSDid_B)(k) != 0) and ([PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed(k) == **True**)

and ([PSDid\_A](#PSDid_A)(k) == 0))

or (([PSDid\_A](#PSDid_A)(k) != 0) and ([PSDzoneStatus\_A](#PSDzoneStatus_A).AllPSDclosed(k) == **True**)

and ([PSDid\_B](#PSDid_B)(k) != 0) and ([PSDzoneStatus\_B](#PSDzoneStatus_B).AllPSDclosed(k) == **True**)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0267], [iTC\_CC\_ATP\_SwHA-0220]

[End]

[iTC\_CC\_ATP-SwRS-0750]

NoDangerforUnexpectedPSDopening，判断在发车时是否PSD为开门状态.

ATP shall determine whether the train starts moving without the aligned PSD closed.

def [NoDangerForUnexpectedPSDopening](#NoDangerForUnexpectedPSDopening)(k):

return (not ([AlignPSDzone\_A](#AlignPSDzone_A)(k) or [AlignPSDzone\_B](#AlignPSDzone_B)(k))

or [AllPSDclosedAndLocked](#AllPSDclosedAndLocked)(k)

or [TrainFilteredStopped](#TrainFilteredStopped)(k)

or not [TrainFilteredStopped](#TrainFilteredStopped)(k-1))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0266], [iTC\_CC\_ATP\_SwHA-0144]

[End]

[iTC\_CC\_ATP-SwRS-0343]

EBforUnexpectedPSDopening，PSD区域内刚发车时PSD门开，则输出EB.

If in charge of the PSD control, ATP shall trigger emergency brake when train just started moving but PSD does not closed.

def [EBforUnexpectedPSDopening](#EBforUnexpectedPSDopening)(k):

return (not [NoDangerForUnexpectedPSDopening](#NoDangerForUnexpectedPSDopening)(k)

and not [InhibitControlPSDstatus](#InhibitControlPSDstatus)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0266], [iTC\_CC-SyAD-1230], [iTC\_CC\_ATP\_SwHA-0144]

[End]

[iTC\_CC\_ATP-SwRS-0803]

NoDangerForPSDnotClosedAndLocked，列车停在PSD区域，且PSD状态为限制时，该值为假；否则，该值为真。

def [NoDangerForPSDnotClosedAndLocked](#NoDangerForPSDnotClosedAndLocked)(k):

return not ([TrainFilteredStopped](#TrainFilteredStopped)(k)

and not [AllPSDclosedAndLocked](#AllPSDclosedAndLocked)(k)

and ([AlignPSDzone\_A](#AlignPSDzone_A)(k) or [AlignPSDzone\_B](#AlignPSDzone_B)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0230]

[End]

[iTC\_CC\_ATP-SwRS-0344]

PBforPSDnotClosedAndLocked，车停在PSD区域内，PSD开，且未限制监控该功能时，要求输出ZVRD。

If one of the statuses of the aligned PSD does not closed when train stopped, ATP shall trigger parking brake.

def [PBforPSDnotClosedAndLocked](#PBforPSDnotClosedAndLocked)(k):

return (not [NoDangerForPSDnotClosedAndLocked](#NoDangerForPSDnotClosedAndLocked)(k)

and not [InhibitControlPSDstatus](#InhibitControlPSDstatus)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0230], [iTC\_CC-SyAD-1221], [iTC\_CC\_ATP\_SwHA-0145]

[End]

[iTC\_CC\_ATP-SwRS-0345]

EBforPBnotAppliedDueToPSD，由于PSD开而施加PB，但是未采到ZVBA。

If ATP has triggered the parking brake for the PSD opening, but it does not applied by the rolling stock, ATP shall trigger the emergency brake.

[EBforPBnotAppliedDueToPSD](#EBforPBnotAppliedDueToPSD)(k)

= (([PBforPSDnotClosedAndLocked](#PBforPSDnotClosedAndLocked)(k) == **True**)

and ([TrainParkingBrakeApplied](#TrainParkingBrakeApplied)(k) != **True**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0233], [iTC\_CC-SyAD-0234], [iTC\_CC-SyAD-0235], [iTC\_CC\_ATP\_SwHA-0146]

[End]

[iTC\_CC\_ATP-SwRS-0346]

PSDopeningCommandValid，开门命令是否在有效期内.

* 如果本周期[PSDopeningCommand](#PSDopeningCommand)为**True**，则设置本周期PSDopeningCommandValid为**True**；
* 否则如果上周期[PSDopeningCommand](#PSDopeningCommand)为**True**而本周期变为**False**，则在接下来的[ATPsetting](#ATPsetting). PSDopeningCommandValidityTime周期内：
* 如果[PSDopeningCommand](#PSDopeningCommand)为**False**，仍保持PSDopeningCommandValid为**True**；
* 如果[PSDopeningCommand](#PSDopeningCommand)为**True**，则当其变为**False**后重新计时。
* 其他情况，设置PSDopeningCommandValid为**False**。

ATP shall monitor the PSD opening command whether valid by the following rules:

* If the PSD opening command is **True** in the current cycle, ATP consider this command is valid;
* Or else:, if the PSD opening command become from **True** to **False**, ATP shall start to counter the cycles:
* If the PSD opening command has become **False** and has lasted more than [ATPsetting](#ATPsetting).PSDopeningCommandValidityTime, then ATP shall consider the PSD opening command as invalid;
* Or else:, ATP still consider the PSD opening command as valid.
* Otherwise, ATP consider the PSD opening command as invalid.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0230], [iTC\_CC\_ATP\_SwHA-0140]

[End]

[iTC\_CC\_ATP-SwRS-0347]

InhibitPSDopeningSupervisedByATP，是否禁止ATP监控发送屏蔽门开启指令时输出PB。

def [InhibitPSDopeningSupervisedByATP](#InhibitPSDopeningSupervisedByATP)(k):

return [Offline.GetInhibitPSDopeningSupervisedByATP](#GetInhibitPSDopeningSupervisedByATP)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1222], [iTC\_CC-SyAD-1305]

[End]

[iTC\_CC\_ATP-SwRS-0348]

PBforPSDopenedAndSupervisedByATP，在PSD开门过程中输出PB

If ATP needs to supervise the PSD opening status, ATP shall trigger parking brake when the PSD opening command is valid.

def [PBforPSDopenedAndSupervisedByATP](#PBforPSDopenedAndSupervisedByATP)(k):

return ([PSDopeningCommandValid](#PSDopeningCommandValid)(k)

and not [InhibitPSDopeningSupervisedByATP](#InhibitPSDopeningSupervisedByATP))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0230], [iTC\_CC-SyAD-1222]

[End]

### Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | | Safety Critical | Observable | | Logical Type | |
| [EBforDepartureWithoutTDCL](#EBforDepartureWithoutTDCL) | √ | | | √ | | BOOLEAN |
| [EBforMovingWithoutTDCL](#EBforMovingWithoutTDCL) | √ | | | √ | | BOOLEAN |
| [EBforPBnotAppliedDueToPSD](#EBforPBnotAppliedDueToPSD) | √ | | | √ | | BOOLEAN |
| [EBforPBnotAppliedDueToTrainDoors](#EBforPBnotAppliedDueToTrainDoors) | √ | | | √ | | BOOLEAN |
| [EBforUnexpectedPSDopening](#EBforUnexpectedPSDopening) | √ | | | √ | | BOOLEAN |
| [InhibitControlPSDstatus](#InhibitControlPSDstatus) | √ | | | √ | | BOOLEAN |
| [InhibitControlTrainDoorsStatus](#InhibitControlTrainDoorsStatus) | √ | | | √ | | BOOLEAN |
| [InhibitProtectionMovingWithoutTDCL](#InhibitProtectionMovingWithoutTDCL) | √ | | | √ | | BOOLEAN |
| [InhibitPSDopeningSupervisedByATP](#InhibitPSDopeningSupervisedByATP) | √ | | | √ | | BOOLEAN |
| [NoDangerForDepartureWithoutTDCL](#NoDangerForDepartureWithoutTDCL) | √ | | | √ | | BOOLEAN |
| [NoDangerForMovingWithoutTDCL](#NoDangerForMovingWithoutTDCL) | √ | | | √ | | BOOLEAN |
| [NoDangerForPSDnotClosedAndLocked](file:///D:\ATP\3-Requirement\ATP-SwRS.docx#NoDangerForPSDnotClosedAndLocked) | √ | | | √ | | BOOLEAN |
| [NoDangerForTrainDoorsNotClosedAndLocked](file:///D:\ATP\3-Requirement\ATP-SwRS.docx#NoDangerForTrainDoorsNotClosedAndLocked) | √ | | | √ | | BOOLEAN |
| [NoDangerForUnexpectedPSDopening](#NoDangerForUnexpectedPSDopening) | √ | | | √ | | BOOLEAN |
| [PBforPSDnotClosedAndLocked](#PBforPSDnotClosedAndLocked) | √ | | | √ | | BOOLEAN |
| [PBforPSDopenedAndSupervisedByATP](#PBforPSDopenedAndSupervisedByATP) | √ | | | √ | | BOOLEAN |
| [PBforTrainDoorsNotClosedAndLocked](#PBforTrainDoorsNotClosedAndLocked) | √ | | | √ | | BOOLEAN |

## F64-Protect Passengers during Emergency Evacuation

本模块用于处理紧急疏散。

This module is for emergency evacuation.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [EnableDoorOpening\_A](#EnableDoorOpening_A) | Internal | F61-Elaborate Door Opening Authorization |
| [EnableDoorOpening\_B](#EnableDoorOpening_B) | Internal | F61-Elaborate Door Opening Authorization |
| [InhibitEmergencyBrake](#InhibitEmergencyBrake) | Internal | F71-Outputs to Rolling-stock |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainMaxSpeed](#TrainMaxSpeed) | Internal | F27-Compute Train Kinematics |
| [ValidTrainKinematic](#ValidTrainKinematic) | Internal | F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [EvacuationNotPossible\_A](#EvacuationNotPossible_A) | √ | √ | BOOLEAN |
| [EvacuationNotPossible\_B](#EvacuationNotPossible_B) | √ | √ | BOOLEAN |
| [TrainDockedInStaion](#TrainDockedInStaion) | √ | √ | BOOLEAN |
| [TrainLeavingStation](#TrainLeavingStation) | √ | √ | BOOLEAN |

### Processing

#### Hold detrainment door closed

对于配有两端驾驶室逃生门的项目，当列车静止或运动学测量失效时，如果端门的紧急逃生手柄被拉下，则ATP应当触发EB，解锁端门，供乘客逃生。

For the project equipped with the detrained door in driving cab, when the train stopped or dynamic measurement invalid, if emergency handle is pulling down, the ATP shall trigger EB, unlock the door for passengers getting out.

[iTC\_CC\_ATP-SwRS-0072]

EmergencyHandleNotPulledEnd1，**END\_1**逃生门未开。如果该项目未配置驾驶室的逃生门，则认为该逃生门未开。其状态来自于项目可配置的列车输入采集。

[EmergencyHandleNotPulledEnd1](#EmergencyHandleNotPulledEnd1) stands for the closed emergency door of **END\_1**. If the train does not allocate with emergency door in the cab, it is certain that the emergency door does not opened.

def [EmergencyHandleNotPulledEnd1](#EmergencyHandleNotPulledEnd1)(k):

return [Offline](#Offline).[GetEmergencyHandleNotPulledEnd1](#GetEmergencyHandleNotPulledEnd1)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0246], [iTC\_CC-SyAD-1003], [iTC\_CC\_ATP\_SwHA-0207]

[End]

[iTC\_CC\_ATP-SwRS-0724]

EmergencyHandleNotPulledEnd2，End\_2逃生门未开。如果该项目未配置驾驶室的逃生门，则认为该逃生门未开。其状态来自于项目可配置的列车输入采集。

[EmergencyHandleNotPulledEnd2](#EmergencyHandleNotPulledEnd2) stands for the closed emergency door of End2. If the train does not allocate with emergency door in the cab, it is certain that the emergency door does not opened.

def [EmergencyHandleNotPulledEnd2](#EmergencyHandleNotPulledEnd2)(k):

return [Offline](#Offline).GetEmergencyHandleNotPulledEnd2(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0246], [iTC\_CC-SyAD-1003], [iTC\_CC\_ATP\_SwHA-0207]

[End]

[iTC\_CC\_ATP-SwRS-0349]

HoldDoorsClosedTrainEnd1，未拉**END\_1**端驾驶室的逃生门紧急手柄，或者车在运动时，锁闭**END\_1**端逃生门。

ATP shall keep hold the train **END\_1** door closed when one of the following conditions fulfilled:

* Train kinematics is valid and the train does not stop;
* Or the emergency handle of **END\_1** is not pulled;

[HoldDoorsClosedTrainEnd1](#HoldDoorsClosedTrainEnd1)(k)

= (([ValidTrainKinematic](#ValidTrainKinematic)(k) and (not [TrainFilteredStopped](#TrainFilteredStopped)(k)))

or [EmergencyHandleNotPulledEnd1](#EmergencyHandleNotPulledEnd1)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0248], [iTC\_CC-SyAD-0249]

[End]

[iTC\_CC\_ATP-SwRS-0350]

HoldDoorsClosedTrainEnd2，未拉**END\_2**端驾驶室的逃生门紧急手柄，或者车在运动时，锁闭**END\_2**端逃生门。

ATP shall keep hold the train **END\_2** door closed when one of the following conditions fulfilled:

* Train kinematics is valid and the train does not stop;
* Or the emergency handle of **END\_2** is not pulled;

[HoldDoorsClosedTrainEnd2](#HoldDoorsClosedTrainEnd2)(k)

= (([ValidTrainKinematic](#ValidTrainKinematic)(k) and (not [TrainFilteredStopped](#TrainFilteredStopped)(k)))

or [EmergencyHandleNotPulledEnd2](#EmergencyHandleNotPulledEnd2)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0248], [iTC\_CC-SyAD-0250]

[End]

[iTC\_CC\_ATP-SwRS-0351]

EBforNotAllTrainEndHoldDoorsClosed，驾驶室逃生门手柄拉下.

If ATP does not hold the train end door, then trigger emergency brake.

[EBforNotAllTrainEndHoldDoorsClosed](#EBforNotAllTrainEndHoldDoorsClosed)(k)

= (not [HoldDoorsClosedTrainEnd1](#HoldDoorsClosedTrainEnd1)(k)

or not [HoldDoorsClosedTrainEnd2](#HoldDoorsClosedTrainEnd2)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0248], [iTC\_CC\_ATP\_SwHA-0148]

[End]

[iTC\_CC\_ATP-SwRS-0738]

EmergencyDetrainDoorLockingEnd1，要求车辆锁闭End1端驾驶室的紧急逃生门。

def [EmergencyDetrainDoorLockingEnd1](#EmergencyDetrainDoorLockingEnd1)(k):

return ([HoldDoorsClosedTrainEnd1](#HoldDoorsClosedTrainEnd1)(k)

or [InhibitEmergencyBrake](#InhibitEmergencyBrake)(k-1))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0249]

[End]

[iTC\_CC\_ATP-SwRS-0739]

EmergencyDetrainDoorLockingEnd2，要求车辆锁闭End2端驾驶室的紧急逃生门。

def [EmergencyDetrainDoorLockingEnd2](#EmergencyDetrainDoorLockingEnd2)(k):

return ([HoldDoorsClosedTrainEnd2](#HoldDoorsClosedTrainEnd2)(k)

or [InhibitEmergencyBrake](#InhibitEmergencyBrake)(k-1))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0250]

[End]

#### Hazardous Evacuation zone

[iTC\_CC\_ATP-SwRS-0273]

EvacuationNotPossible\_A，禁止A侧逃生。

* 初始化或[TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)为**False**时，默认[EvacuationNotPossible\_A](#EvacuationNotPossible_A)为**False**。
* 否则，当车身定位（即从车尾最小定位到车头最大定位）与禁止逃生区（线路地图[TrackMap](#TrackMap)中由一对方向相反的**SGL\_HAZAR\_EVAC\_ZONE**奇点组成）范围有交集，且根据Table 5‑12判断为A侧时，设置[EvacuationNotPossible\_A](#EvacuationNotPossible_A)为**True**：
* 其他情况，设置[EvacuationNotPossible\_A](#EvacuationNotPossible_A)为**False**。

ATP shall consider the evacuation is not possible on side-A when there are intersection between the range of train locations and the hazardous evacuation zone of the side-A:

* In initialization or not [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath), set the [EvacuationNotPossible\_A](#EvacuationNotPossible_A) as **False**;
* Or else, if there are intersection between the range of train locations (from the minimum train tail to the maximum train head) and the vital passenger exchange zone (composing by a pair of **SGL\_HAZAR\_EVAC\_ZONE** singularities with opposite direction in the train map), and the train door side A correspond to the EVAC according with Table 5‑12, set [EvacuationNotPossible\_A](#EvacuationNotPossible_A) as **True**.
* Otherwise, set [EvacuationNotPossible\_A](#EvacuationNotPossible_A) as **False**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0244], [iTC\_CC\_ATP\_SwHA-0112]

[End]

[iTC\_CC\_ATP-SwRS-0274]

EvacuationNotPossible\_B，禁止B侧逃生。

* 初始化或[TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath)为**False**时，默认[EvacuationNotPossible\_B](#EvacuationNotPossible_B)为**False**。
* 否则，当车身定位（即从车尾最小定位到车头最大定位）与禁止逃生区（线路地图[TrackMap](#TrackMap)中由一对方向相反的**SGL\_HAZAR\_EVAC\_ZONE**奇点组成）范围有交集，且根据Table 5‑12判断为B侧时，设置[EvacuationNotPossible\_B](#EvacuationNotPossible_B)为**True**。
* 其他情况，设置[EvacuationNotPossible\_B](#EvacuationNotPossible_B)为**False**。

ATP shall consider the evacuation is not possible on side-B when there are intersection between the range of train locations and the hazardous evacuation zone of the side-B:

* In initialization or not [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath), set the [EvacuationNotPossible\_B](#EvacuationNotPossible_B) as **False**;
* Or else, if there are intersection between the range of train locations (from the minimum train tail to the maximum train head) and the vital passenger exchange zone (composing by a pair of **SGL\_HAZAR\_EVAC\_ZONE** singularities with opposite direction in the train map), and the train door side B correspond to the EVAC according with Table 5‑12, set [EvacuationNotPossible\_B](#EvacuationNotPossible_B) as **True**.
* Otherwise, set [EvacuationNotPossible\_B](#EvacuationNotPossible_B) as **False**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0244], [iTC\_CC\_ATP\_SwHA-0112]

[End]

#### Hold train side doors closed

当列车车速大于项目配置数据时，ATP应当要求列车锁闭车厢两侧的车门；只有当车速低于项目配置数据的值，且该侧车门方向允许逃生（由线路上的禁止逃生区限制）或者项目没有配置驾驶室的端门时，ATP才允许列车解除该侧车门的锁闭请求。

When the train speed is higher than project configuration data, ATP shall request the train to close and lock on both sides of the door. Only when the speed is lower than the value of the project configuration data, and this side permits for the escape (is determined by restricted escape area in the lines), or there is no end door configured in the cab, ATP shall allow to release the door locking request.

[iTC\_CC\_ATP-SwRS-0352]

HoldDoorsClosed\_A，A侧车门锁闭.

The conditions ATP determining the [HoldDoorsClosed\_A](#HoldDoorsClosed_A) show as following ARDL:

if (Initialization

or ([ValidTrainKinematic](#ValidTrainKinematic)(k) != **True**))

[HoldDoorsClosed\_A](#HoldDoorsClosed_A) = **False**

elif ([TrainMaxSpeed](#TrainMaxSpeed)(k) > [ATPsetting](#ATPsetting).DoorTrainLockingSpeed)

[HoldDoorsClosed\_A](#HoldDoorsClosed_A) = **True**

elif ([TrainMaxSpeed](#TrainMaxSpeed)(k) <= [ATPsetting](#ATPsetting).DoorTrainUnlockingSpeed)

[HoldDoorsClosed\_A](#HoldDoorsClosed_A)(k)

= (([EvacuationNotPossible\_A](#EvacuationNotPossible_A)(k) and (not [EvacuationNotPossible\_B](#EvacuationNotPossible_B)(k)))

or ([EvacuationNotPossible\_A](#EvacuationNotPossible_A)(k) and [EvacuationNotPossible\_B](#EvacuationNotPossible_B)(k)

and [ATPsetting](#ATPsetting).EvacuationTrainEnd))

else:

[HoldDoorsClosed\_A](#HoldDoorsClosed_A) = [HoldDoorsClosed\_A](#HoldDoorsClosed_A)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0240], [iTC\_CC-SyAD-0244], [iTC\_CC-SyAD-0245], [iTC\_CC\_ATP\_SwHA-0149]

[End]

[iTC\_CC\_ATP-SwRS-0353]

HoldDoorsClosed\_B，B侧车门锁闭

The conditions ATP determining the [HoldDoorsClosed\_B](#HoldDoorsClosed_B) show as following ARDL :

if (Initialization

or ([ValidTrainKinematic](#ValidTrainKinematic)(k) != **True**))

[HoldDoorsClosed\_B](#HoldDoorsClosed_B) = **False**

elif ([TrainMaxSpeed](#TrainMaxSpeed)(k) > [ATPsetting](#ATPsetting).DoorTrainLockingSpeed)

[HoldDoorsClosed\_B](#HoldDoorsClosed_B) = **True**

elif ([TrainMaxSpeed](#TrainMaxSpeed)(k) <= [ATPsetting](#ATPsetting).DoorTrainUnlockingSpeed)

[HoldDoorsClosed\_B](#HoldDoorsClosed_B)(k)

= (([EvacuationNotPossible\_B](#EvacuationNotPossible_B)(k) and (not [EvacuationNotPossible\_A](#EvacuationNotPossible_A)(k)))

or ([EvacuationNotPossible\_A](#EvacuationNotPossible_A)(k) and [EvacuationNotPossible\_B](#EvacuationNotPossible_B)(k)

and [ATPsetting](#ATPsetting).EvacuationTrainEnd))

else:

[HoldDoorsClosed\_B](#HoldDoorsClosed_B) = [HoldDoorsClosed\_B](#HoldDoorsClosed_B)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0240], [iTC\_CC-SyAD-0244], [iTC\_CC-SyAD-0245], [iTC\_CC\_ATP\_SwHA-0149]

[End]

#### Evacuation requirement

当列车在站内发车，或者未完全出站时，如果乘客拉下了两侧车门的紧急逃生手柄，则ATP应当输出EB，使得列车停止；但如果列车已在区间运行，乘客拉下紧急逃生手柄，则ATP不应当输出EB。

When the train is just leaving from the platform or stops out of the station, if the passenger pulls down the emergency handle, ATP shall trigger EB to stop the train. If the train is running on the interval region, when the passenger pulls down the handle, the ATP shall not output EB.

[iTC\_CC\_ATP-SwRS-0354]

TrainDockedInStation，根据开门授权条件判断是否车停在站内。

ATP shall determine whether the train has docked in the station correctly according to conditions of train stopping and doors opening enable.

def [TrainDockedInStation](#TrainDockedInStation)(k):

return ([TrainFilteredStopped](#TrainFilteredStopped)(k)

and ([EnableDoorOpening\_A](#EnableDoorOpening_A)(k)

or [EnableDoorOpening\_B](#EnableDoorOpening_B)(k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0229]

[End]

[iTC\_CC\_ATP-SwRS-0355]

TrainLeavingStation，判断是否在离站过程中。

* 从[TrainDockedInStation](#TrainDockedInStation)由**True**变为**False**开始，如果列车测速有效，累加[MaximumTrainMotion](#MaximumTrainMotion)距离：
* 如果其绝对值在[**0**, [ATPsetting](#ATPsetting).EvacuationStationAreaLength]范围内，则设置TrainLeavingStation为**True**；否则为**False**。
* 即如果列车出站后又倒车回到上述范围内，也应认为是TrainLeavingStation。
* 如果列车运动学无效，则设置该值为**False**并清除累加距离。

The train is said to be leaving the station:

* if since last time train has been detected docked in station ([TrainDockedInStation](#TrainDockedInStation)), the cumulated of the absolute value of [MaximumTrainMotion](#MaximumTrainMotion) is in the range [**0**, [ATPsetting](#ATPsetting).EvacuationStationAreaLength] and no train kinematic invalidation occurs.
* or else, if the train kinematics is invalid, ATP shall set [TrainLeavingStation](#TrainLeavingStation) as **False** and clear the cumulated distance.

def [TrainLeavingStation](#TrainLeavingStation)(k):

if (Initialization

or not [ValidTrainKinematic](#ValidTrainKinematic)(k)):

TrainHasDockedInStation = **False**

LeavingStationDistance = **0**

return **False**

elif ([TrainDockedInStation](#TrainDockedInStation)(k)):

TrainHasDockedInStation = **True**

LeavingStationDistance = **0**

return **False**

elif (not TrainHasDockedInStation(k)):

LeavingStationDistance = **0**

return **False**

else:

LeavingStationDistance = LeavingStationDistance(k-1) + [MaximumTrainMotion](#MaximumTrainMotion)(k)

return ([abs](#abs)(LeavingStationDistance) <= [ATPsetting](#ATPsetting).EvacuationStationAreaLength)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0271], [iTC\_CC-SyAD-1231]

[End]

[iTC\_CC\_ATP-SwRS-0071]

EmergencyHandleNotPulledSide侧向的紧急手柄未落下。其状态来自于项目可配置的列车输入采集。

EmergencyHandleNotPulledSid shows that the emergency handles is not pulled down.

def [EmergencyHandleNotPulledSide](#EmergencyHandleNotPulledSide)(k):

return [Offline](#Offline).[GetEmergencyHandleNotPulledSide](#GetEmergencyHandleNotPulledSide)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0269], [iTC\_CC-SyAD-1003], [iTC\_CC\_ATP\_SwHA-0207]

[End]

[iTC\_CC\_ATP-SwRS-0356]

EvacuationWhileLeavingStation，未完全出站时丢失车门状态则EB.

If the train is just leaving the station and the side doors emergency handles are pulled, ATP shall require [EvacuationWhileLeavingStation](#EvacuationWhileLeavingStation).

[EvacuationWhileLeavingStation](#EvacuationWhileLeavingStation)(k)

= (([EmergencyHandleNotPulledSide](#EmergencyHandleNotPulledSide)(k) != **True**)

and ([TrainLeavingStation](#TrainLeavingStation)(k) == **True**)

and ([TrainFilteredStopped](#TrainFilteredStopped)(k) != **True**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0271], [iTC\_CC-SyAD-1231], [iTC\_CC\_ATP\_SwHA-0150]

[End]

[iTC\_CC\_ATP-SwRS-0357]

EvacuationWithTrainStopped，非开门区，停车且乘客紧急手柄拉下

If the train does not stop on the doors opening enable area and the side doors emergency handles pulled, ATP shall require [EvacuationWithTrainStopped](#EvacuationWithTrainStopped).

[EvacuationWithTrainStopped](#EvacuationWithTrainStopped)(k)

= (([EmergencyHandleNotPulledSide](#EmergencyHandleNotPulledSide)(k) != **True**)

and ([TrainFilteredStopped](#TrainFilteredStopped)(k) == **True**)

and not [EnableDoorOpening\_A](#EnableDoorOpening_A)(k)

and not [EnableDoorOpening\_B](#EnableDoorOpening_B)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0274], [iTC\_CC\_ATP\_SwHA-0151]

[End]

[iTC\_CC\_ATP-SwRS-0726]

InhibitProtectionEvacuationInDistance，在离站时禁止监控逃生手柄状态

def [InhibitProtectionEvacuationInDistance](#InhibitProtectionEvacuationInDistance)(k):

return [Offline.GetInhibitProtectionEvacuationInDistance](#GetInhibitProtectionEvacuationInDistance)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1231], [iTC\_CC-SyAD-1302]

[End]

[iTC\_CC\_ATP-SwRS-0727]

InhibitProtectionEvacuationWithStop，在站间停车时禁止监控逃生手柄状态。

def [InhibitProtectionEvacuationWithStop](#InhibitProtectionEvacuationWithStop)(k):

return [Offline](#Offline).[GetInhibitProtectionEvacuationWithStop](#GetInhibitProtectionEvacuationWithStop)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1232], [iTC\_CC-SyAD-1301]

[End]

[iTC\_CC\_ATP-SwRS-0358]

EBforEvacuationWhileTrainLeavingStation，出站时的逃生请求EB.

If the train leaving station evacuation has been required, ATP shall trigger the emergency brake.

def [EBforEvacuationWhileTrainLeavingStation](#EBforEvacuationWhileTrainLeavingStation)(k):

return ([EvacuationWhileLeavingStation](#EvacuationWhileLeavingStation)(k)

and not [InhibitProtectionEvacuationInDistance](#InhibitProtectionEvacuationInDistance)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0271], [iTC\_CC-SyAD-1231], [iTC\_CC\_ATP\_SwHA-0150]

[End]

[iTC\_CC\_ATP-SwRS-0748]

EBforEvacuationWithTrainStopped，站间停车时的逃生请求EB.

If the train stopped evacuation has been required, ATP shall trigger the emergency brake.

def [EBforEvacuationWithTrainStopped](#EBforEvacuationWithTrainStopped)(k):

return ([EvacuationWithTrainStopped](#EvacuationWithTrainStopped)(k)

and not [InhibitProtectionEvacuationWithStop](#InhibitProtectionEvacuationWithStop)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0274], [iTC\_CC-SyAD-1232], [iTC\_CC\_ATP\_SwHA-0151]

[End]

### Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | | Safety Critical | | Observable | Logical Type | |
| [EBforEvacuationWhileTrainLeavingStation](#EBforEvacuationWhileTrainLeavingStation) | √ | | √ | | | BOOLEAN |
| [EBforEvacuationWithTrainStopped](#EBforEvacuationWithTrainStopped) | √ | | √ | | | BOOLEAN |
| [EBforNotAllTrainEndHoldDoorsClosed](#EBforNotAllTrainEndHoldDoorsClosed) | √ | | √ | | | BOOLEAN |
| [EmergencyDetrainDoorLockingEnd1](#EmergencyDetrainDoorLockingEnd1) | √ | | √ | | | BOOLEAN |
| [EmergencyDetrainDoorLockingEnd2](#EmergencyDetrainDoorLockingEnd2) | √ | | √ | | | BOOLEAN |
| [EmergencyHandleNotPulledSide](#EmergencyHandleNotPulledSide) | √ | | √ | | | BOOLEAN |
| [EmergencyHandleNotPulledEnd1](#EmergencyHandleNotPulledEnd1) | √ | | √ | | | BOOLEAN |
| [EmergencyHandleNotPulledEnd2](#EmergencyHandleNotPulledEnd2) | √ | | √ | | | BOOLEAN |
| [EvacuationWhileLeavingStation](#EvacuationWhileLeavingStation) | √ | | √ | | | BOOLEAN |
| [EvacuationWithTrainStopped](#EvacuationWithTrainStopped) | √ | | √ | | | BOOLEAN |
| [HoldDoorsClosedTrainEnd1](#HoldDoorsClosedTrainEnd1) | √ | | √ | | | BOOLEAN |
| [HoldDoorsClosedTrainEnd2](#HoldDoorsClosedTrainEnd2) | √ | | √ | | | BOOLEAN |
| [HoldDoorsClosed\_A](#HoldDoorsClosed_A) | √ | | √ | | | BOOLEAN |
| [HoldDoorsClosed\_B](#HoldDoorsClosed_B) | √ | | √ | | | BOOLEAN |
| [InhibitProtectionEvacuationInDistance](#InhibitProtectionEvacuationInDistance) | √ | | √ | | | BOOLEAN |
| [InhibitProtectionEvacuationWithStop](#InhibitProtectionEvacuationWithStop) | √ | | √ | | | BOOLEAN |

## F7-Generate Output Orders

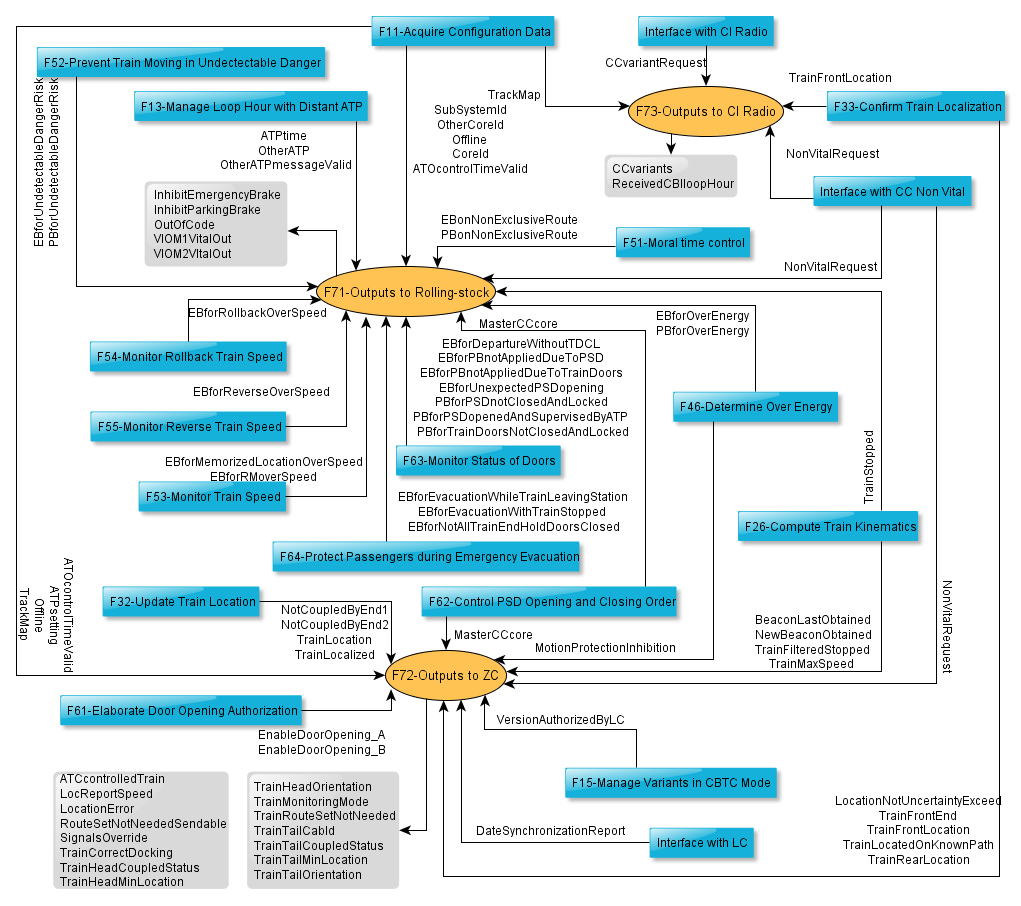


Figure 5‑21 SART modeling of function F7

## F71-Outputs to Rolling-stock

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATOcontrolTimeValid](#ATOcontrolTimeValid) | Internal | F11-Acquire Configuration Data |
| [ATPtime](#ATPtime) | Internal | F13-Manage Loop Hour with Distant ATP |
| [CoreId](#CoreId) | Internal | F11-Acquire Configuration Data |
| [EBforDepartureWithoutTDCL](#EBforDepartureWithoutTDCL) | Internal | F63-Monitor Status of Doors |
| [EBforEvacuationWhileTrainLeavingStation](#EBforEvacuationWhileTrainLeavingStation) | Internal | F64-Protect Passengers during Emergency Evacuation |
| [EBforEvacuationWithTrainStopped](#EBforEvacuationWithTrainStopped) | Internal | F64-Protect Passengers during Emergency Evacuation |
| [EBforMemorizedLocationOverSpeed](#EBforMemorizedLocationOverSpeed) | Internal | F53-Monitor Train Speed |
| [EBforMovingWithoutTDCL](#EBforMovingWithoutTDCL) | Internal | F63-Monitor Status of Doors |
| [EBforNotAllTrainEndHoldDoorsClosed](#EBforNotAllTrainEndHoldDoorsClosed) | Internal | F64-Protect Passengers during Emergency Evacuation |
| [EBforOverEnergy](#EBforOverEnergy) | Internal | F46-Determine Over Energy |
| [EBforPBnotAppliedDueToPSD](#EBforPBnotAppliedDueToPSD) | Internal | F63-Monitor Status of Doors |
| [EBforPBnotAppliedDueToTrainDoors](#EBforPBnotAppliedDueToTrainDoors) | Internal | F63-Monitor Status of Doors |
| [EBforReverseOverSpeed](#EBforReverseOverSpeed) | Internal | F55-Monitor Reverse Train Speed |
| [EBforRMoverSpeed](#EBforRMoverSpeed) | Internal | F53-Monitor Train Speed |
| [EBforRollbackOverSpeed](#EBforRollbackOverSpeed) | Internal | F54-Monitor Rollback Train Speed |
| [EBforUnexpectedPSDopening](#EBforUnexpectedPSDopening) | Internal | F63-Monitor Status of Doors |
| [EBforUndetectableDangerRisk](#EBforUndetectableDangerRisk) | Internal | F52-Prevent Train Moving in Undetectable Danger |
| [EBonNonExclusiveRoute](#EBonNonExclusiveRoute) | Internal | F51-Moral Time |
| [LatestVIOM1LoopHourVIOM](#LatestVIOM1LoopHourVIOM) | Internal | F12-Manage Train Status |
| [LatestVIOM2LoopHourVIOM](#LatestVIOM2LoopHourVIOM) | Internal | F12-Manage Train Status |
| [MasterCCcore](#MasterCCcore) | Internal | F62-Control PSD Opening and Closing Order |
| [MatchRebootCondition](#MatchRebootCondition) | Internal | F12-Manage Train Status |
| [NonVitalRequest](#NonVitalRequest) | External | Interface with CC Non Vital |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [OtherATP](#OtherATP) | Internal | F13-Manage Loop Hour with Distant ATP |
| [OtherATPmessageValid](#OtherATPmessageValid) | Internal | F13-Manage Loop Hour with Distant ATP |
| [OtherCoreId](#OtherCoreId) | Internal | F11-Acquire Configuration Data |
| [PBforOverEnergy](#PBforOverEnergy) | Internal | F46-Determine Over Energy |
| [PBforPSDnotClosedAndLocked](#PBforPSDnotClosedAndLocked) | Internal | F63-Monitor Status of Doors |
| [PBforPSDopenedAndSupervisedByATP](#PBforPSDopenedAndSupervisedByATP) | Internal | F63-Monitor Status of Doors |
| [PBforTrainDoorsNotClosedAndLocked](#PBforTrainDoorsNotClosedAndLocked) | Internal | F63-Monitor Status of Doors |
| [PBforUndetectableDangerRisk](#PBforUndetectableDangerRisk) | Internal | F52-Prevent Train Moving in Undetectable Danger |
| [PBonNonExclusiveRoute](#PBonNonExclusiveRoute) | Internal | F51-Moral Time |
| [SafeTimerFailed](#SafeTimerFailed) | Internal | F82-Manage Vital Time |
| [SubSystemId](#SubSystemId) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [ApproachableSignalOverrun](#ApproachableSignalOverrun) | √ | √ | BOOLEAN |
| [CCworkOvertime](#CCworkOvertime) | √ | √ | BOOLEAN |
| [EBforOperationRequest](#EBforOperationRequest) | √ | √ | BOOLEAN |
| [IncompatibleDistantATP](#IncompatibleDistantATP) | √ | √ | BOOLEAN |
| [PBforOperationRequest](#PBforOperationRequest) | √ | √ | BOOLEAN |
| [TrainEmergencyBrakeRequested](#TrainEmergencyBrakeRequested) | √ | √ | BOOLEAN |
| [TrainParkingBrakeRequested](#TrainParkingBrakeRequested) | √ | √ | BOOLEAN |
| [VIOM1OutNotDisabled](#VIOM1OutNotDisabled) | √ | √ | BOOLEAN |
| [VIOM2OutNotDisabled](#VIOM2OutNotDisabled) | √ | √ | BOOLEAN |

### Processing

#### Parking brake

ATP控制车辆是否输出停车制动，出于安全考虑，在要求输出停车制动时将该输出端口设置为限制状态，不输出停车制动时将该端口置为允许状态。

ATP shall control the parking brake of the train. Due to the safety oriented, ATP shall set the output port as restricted status when outputting the parking brake order, and set the port as permissible status when the ATP does not send parking brake.

[iTC\_CC\_ATP-SwRS-0133]

PBforOperationalRequest，来自CCNV的ZVRD输出请求

PBforOperationalRequest stands for the ZVRD output order from CCNV.

if ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k) == **True**)

[PBforOperationalRequest](#PBforOperationalRequest)(k)

= not [NonVitalRequest](#NonVitalRequest).VitalParkingBrakingNotRequested(k)

else:

[PBforOperationalRequest](#PBforOperationalRequest) = **True**

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0068], [iTC\_CC-SyAD-1044], [iTC\_CC-SyAD-0231]

[End]

[iTC\_CC\_ATP-SwRS-0359]

TrainParkingBrakeRequested，判断本周期是否需要施加停车制动。条件如下：

* 由于moral time 导致需要输出停车制动；
* 或者，由于超能导致需要输出停车制动；
* 或者，由于在PSD区域内车门未关闭而导致需要输出停车制动；
* 或者，由于NUDE导致需要输出停车制动；
* 或者，CCNV请求需要输出停车制动；
* 或者，由于PSD未关闭而导致需要输出停车制动
* 或者，本周期已请求EB输出。

[TrainParkingBrakeRequested](#TrainParkingBrakeRequested), determine whether to apply parking brake. This variable shall be **True** when one of the following conditions met:

* Train is in front of a possibly non-exclusive route,
* Synthesis of speed constraints on the train implies that it is not allowed to move anymore. Any movement may lead to an hazardous situation,
* Train is located on a passenger exchange area with PSD and train doors are not proven closed and locked,
* Train is located on a passenger exchange area with PSD and PSD are not proven closed and locked,
* There is a possibility of undetectable dangers,
* An operational parking brake is requested,
* The PSD are opened and are under the supervision of ATP,
* The EB has been requested in this cycle.

[TrainParkingBrakeRequested](#TrainParkingBrakeRequested) = ([PBonNonExclusiveRoute](#PBonNonExclusiveRoute)(k)

or [PBforOverEnergy](#PBforOverEnergy)(k)

or [PBforTrainDoorsNotClosedAndLocked](#PBforTrainDoorsNotClosedAndLocked)(k)

or [PBforPSDnotClosedAndLocked](#PBforPSDnotClosedAndLocked)(k)

or [PBforUndetectableDangerRisk](#PBforUndetectableDangerRisk)(k)

or [PBforOperationalRequest](#PBforOperationalRequest)(k)

or [PBforPSDopenedAndSupervisedByATP](#PBforPSDopenedAndSupervisedByATP)(k)

or not [InhibitEmergencyBrake](#InhibitEmergencyBrake)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0230], [iTC\_CC\_ATP\_SwHA-0153]

[End]

[iTC\_CC\_ATP-SwRS-0360]

InhibitParkingBrake，当前不施加停车制动。

[InhibitParkingBrake](#InhibitParkingBrake)，ATP software do not apply the parking brake.

[InhibitParkingBrake](#InhibitParkingBrake) = not [TrainParkingBrakeRequested](#TrainParkingBrakeRequested)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0230]

[End]

#### Emergency brake

ATP使用控制车辆是否输出紧急制动，出于安全考虑，在要求输出紧急制动时将该输出端口设置为限制状态，不输出停车制动时将该端口置为允许状态。

ATP shall control the emergency brake of the train. Due to the safety oriented, ATP will set the output port as restricted status when outputting the emergency brake order, and set the port as permissive status when the ATP does not send trigger the brake.

[iTC\_CC\_ATP-SwRS-0556]

IncompatibleDistantATP，判断本ATP与冗余ATP之间的[Coreld](#Coreld)和[SubSystemID](#SubSystemID)是否相匹配。当初始化，冗余ATP信息不可用，或者冗余ATP读取的Dataplug中的SSID与本ATP相一致而[Coreld](#Coreld)不一致时，认为两端ATP相互匹配；否则，ATP将触发紧急制动。

The [Coreld](#Coreld) and [SubSystemID](#SubSystemID) of the ATP and redundant ATP need to compare for the consistency, which records in [IncompatibleDistantATP](#IncompatibleDistantATP). In initialization, the message from redundant ATP cannot be used. On the other hand, when SubSystemID in the Dataplug read by redundant ATP is the same, but the Coreld is different, both ATP regards as consistency. Otherwise, ATP would trigger emergency brake.

def [IncompatibleDistantATP](#IncompatibleDistantATP)(k):

if (Initialization

or not [OtherATPmessageValid](#OtherATPmessageValid)(k)):

return **False**

elif ([OtherCoreId](#OtherCoreId)(k) != [OtherATP](#OtherATP)(k).[CoreId](#CoreId)

or [SubSystemId](#SubSystemId)(k) != [OtherATP](#OtherATP)(k).CC\_SSID):

return **True**

else:

return [IncompatibleDistantATP](#IncompatibleDistantATP)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0964], [iTC\_CC\_ATP\_SwHA-0224]

[End]

[iTC\_CC\_ATP-SwRS-0271]

ApproachableSignalOverrun，本周期列车车头最大定位是否冒进可接近信号机。

* 当满足以下所有条件时，ATP认为列车冒进了可接近信号机，则设置ApproachableSignalOverrun为**True**；
* 当前未选择[MotionProtectionInhibition](#MotionProtectionInhibition)；
* 上周期ApproachableSignalOverrun为**False**；
* 本周期列车位移[MaximumTrainMotion](#MaximumTrainMotion)向激活的驾驶室方向运行；
* 本周期列车车头最大定位[TrainFrontLocation](#TrainFrontLocation)经过了一个带OVERLAP的信号机奇点；
* 该信号机变量状态是限制而其所带Overlap已建立（Overlap状态建立的条件见Table 5‑9）。
* 否则，设置ApproachableSignalOverrun为**False**。

[ApproachableSignalOverrun](#ApproachableSignalOverrun), ATP shall determine whether the location of maximum train head overruns an approachable signal with overlap established.

* When all of the following conditions fulfilled, ATP considers the train has overrun a restricted signal in this cycle, and set [ApproachableSignalOverrun](#ApproachableSignalOverrun) as **True**.
* not [MotionProtectionInhibition](#MotionProtectionInhibition);
* And [ApproachableSignalOverrun](#ApproachableSignalOverrun) was **False** at the last cycle;
* And the moving direction in current cycle is toward on the train front end;
* And the maximum location of train front end passes the position of the signal with overlap attribute in this cycle;
* And the variants status of the signal is overlap established(refer to Table 5‑9).
* Otherwise, ATP set [ApproachableSignalOverrun](#ApproachableSignalOverrun) as **False**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0318] , [iTC\_CC\_ATP\_SwHA-0110]

[End]

[iTC\_CC\_ATP-SwRS-0132]

EBforOperationalRequest，来自CCNV的EB输出请求

ATP shall trigger emergency brake according to CCNV‘s operational emergency brake request.

if ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k) == **True**)

[EBforOperationalRequest](#EBforOperationalRequest)(k) = not [NonVitalRequest](#NonVitalRequest).EmergencyBrakingNotRequested(k)

else:

[EBforOperationalRequest](#EBforOperationalRequest) = **True**

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0068], [iTC\_CC-SyAD-1044]

[End]

[iTC\_CC\_ATP-SwRS-0361]

TrainEmergencyBrakeRequested，判断本周期是否需要施加EB。

ATP shall control emergency brake output according following emergency braking requests from control functions:

* moral-time control function has detected an hazardous situation (route exclusivity violation);
* train speed is no longer compliant with respect of whole speed restriction of guide way;
* an approachable speed limit has been over-run (RM speed limit or memorized location speed limit);
* an over-speed in reverse direction of travel has been detected;
* an emergency evacuation is required for passengers;
* train departure with not all doors closed and locked has been detected;
* the train starts to move on a PSD zone which status is not "all PSD proven closed and locked";
* train has moved although there are potential undetectable dangers;
* an operational emergency braking has been requested by CC-Non Vital;
* train end doors are not closed and locked;
* not all doors closed and locked has been detected on a PSD zone and parking brake is not applied,
* not all PSD closed and locked has been detected on a PSD zone and parking brake is not applied;
* the approachable signal is overrun;
* the VLE-2 safe timer failed;
* the information of Dataplug in both ends of cab is inconsistent.

[TrainEmergencyBrakeRequested](#TrainEmergencyBrakeRequested)(k)

= [EBonNonExclusiveRoute](#EBonNonExclusiveRoute)(k)

or [EBforOverEnergy](#EBforOverEnergy)(k)

or [EBforRMoverSpeed](#EBforRMoverSpeed)(k)

or [EBforMemorizedLocationOverSpeed](#EBforMemorizedLocationOverSpeed)(k)

or [EBforRollbackOverSpeed](#EBforRollbackOverSpeed)(k)

or [EBforReverseOverSpeed](#EBforReverseOverSpeed)(k)

or [EBforEvacuationWhileTrainLeavingStation](#EBforEvacuationWhileTrainLeavingStation)(k)

or [EBforEvacuationWithTrainStopped](#EBforEvacuationWithTrainStopped)(k)

or [[EBforDepartureWithoutTDCL](#EBforDepartureWithoutTDCL)](#DepartureWithNoTDCL)(k)

or [EBforMovingWithoutTDCL](#EBforMovingWithoutTDCL)(k)

or [EBforUnexpectedPSDopening](#EBforUnexpectedPSDopening)(k)

or [EBforUndetectableDangerRisk](#EBforUndetectableDangerRisk)(k)

or [EBforOperationalRequest](#EBforOperationalRequest)(k)

or [EBforNotAllTrainEndHoldDoorsClosed](#EBforNotAllTrainEndHoldDoorsClosed)(k)

or [EBforPBnotAppliedDueToTrainDoors](#EBforPBnotAppliedDueToTrainDoors)(k)

or [EBforPBnotAppliedDueToPSD](#EBforPBnotAppliedDueToPSD)(k)

or [ApproachableSignalOverrun](#ApproachableSignalOverrun)(k)

or [SafeTimerFailed](#SafeTimerFailed)(k)

or [IncompatibleDistantATP](#IncompatibleDistantATP)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0136], [iTC\_CC-SyAD-0248], [iTC\_CC-SyAD-0266], [iTC\_CC-SyAD-0316], [iTC\_CC-SyAD-0329], [iTC\_CC-SyAD-0331], [iTC\_CC-SyAD-0352], [iTC\_CC-SyAD-0362], [iTC\_CC-SyAD-0964], [iTC\_CC\_ATP\_SwHA-0143], [iTC\_CC\_ATP\_SwHA-0146], [iTC\_CC\_ATP\_SwHA-0148], [iTC\_CC\_ATP\_SwHA-0154], [iTC\_CC-SyAD-0137], [iTC\_CC-SyAD-0271], [iTC\_CC-SyAD-0274], [iTC\_CC-SyAD-0361]

[End]

[iTC\_CC\_ATP-SwRS-0362]

InhibitEmergencyBrake，输出和缓解EB的条件

If an emergency braking request ordered by a control function, ATP shall not inhibit emergency brake until train filtered-stop reached.

ATP shall inhibit emergency brake if and only if train detected at filtered stop and there is no emergency braking request from control functions.

if ([InhibitEmergencyBrake](#InhibitEmergencyBrake)(k-1) == **True**)

[InhibitEmergencyBrake](#InhibitEmergencyBrake) = not [TrainEmergencyBrakeRequested](#TrainEmergencyBrakeRequested)(k)

elif (([InhibitEmergencyBrake](#InhibitEmergencyBrake)(k-1) == **False**)

and ([TrainFilteredStopped](#TrainFilteredStopped)(k) == **True**))

[InhibitEmergencyBrake](#InhibitEmergencyBrake) = not [TrainEmergencyBrakeRequested](#TrainEmergencyBrakeRequested)(k)

else:

[InhibitEmergencyBrake](#InhibitEmergencyBrake) = [InhibitEmergencyBrake](#InhibitEmergencyBrake)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0362], [iTC\_CC-SyAD-0364], [iTC\_CC\_ATP\_SwHA-0155]

[End]

#### Vital outputs

对于不同项目，ATP内部计算的输出变量所对应的端口号是可配置的，该对应关系在项目配置数据中设定。

For different project, the internal calculating output ports of ATP are configurable, which defined in the project vital settings.

[iTC\_CC\_ATP-SwRS-0546]

CCworkOvertime，监控CC是否连续工作超过**MAX\_RESET\_TIME**时间(该时间小于**MAX\_ATP\_LOOP\_HOUR**)。如果CC运行超过**MAX\_RESET\_TIME**时间，则ATP需将所有对VIOM输出的端口置为限制状态。

ATP shall monitor the CC continuous work time. If the CC is running more than **MAX\_RESET\_TIME** (the value is far less than **MAX\_ATP\_LOOP\_HOUR**), the ATP shall set all output to VIOM as restricted status.

def [CCworkOvertime](#CCworkOvertime)(k):

return (([CoreId](#CoreId)(k) is **END\_1**

and (([ATPtime](#ATPtime)(k) - **CC1\_INIT\_TIME**) > MAX\_RESET\_TIME))

or ([CoreId](#CoreId)(k) is **END\_2**

and (([ATPtime](#ATPtime)(k) - **CC2\_INIT\_TIME**) > MAX\_RESET\_TIME)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0361], [iTC\_CC\_ATP\_SwHA-0217]

[End]

[iTC\_CC\_ATP-SwRS-0584]

VIOM1OutNotDisabled，VIOM2OutNotDisabled，CCNV请求“非禁止安全输出”。

* 当来自CCNV的消息无效时，应设置CCNV请求的“非禁止安全输出”为限制状态；
* 否则，根据CCNV发送的状态字进行设置。

Whether CCNV request the channel of VIOM shall be disabled or not.

def [VIOM1OutNotDisabled](#VIOM1OutNotDisabled)(port, k):

return ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k)

and [NonVitalRequest](#NonVitalRequest)(k).Viom1[port])

def [VIOM2OutNotDisabled](#VIOM2OutNotDisabled)(port, k):

return ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k)

and [NonVitalRequest](#NonVitalRequest)(k).Viom2[port])

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1132]

[End]

**NOTES**：

只有当CCNV“非禁止”某路安全输出为**True**（即允许某路安全输出）时，ATP才能根据自身的运行结果设置该路输出；而如果CCNV“非禁止”该路输出为**False**时，则ATP应设置该路输出为限制状态。就是说，CCNV可以通过设置“非禁止”某路安全输出为**False**，来强制某路输出为限制状态（导向安全侧）；但无法强制任何一路安全输出为**True**（导向危险侧），因此该功能不会导致安全问题。该功能主要在CCNV进行组合测试时使用。

When some vital outputs are "not disabled" by CCNV, ATP shall set their values as commands calculated by ATP; and if CCNV disabled these outputs channel, ATP shall set their values as restriction. That is to say, CCNV can coerce some output channels as restricted status (to the safe side), but cannot coerce them as permissive (to the hazardous side). This function normally used for the combined test.

[iTC\_CC\_ATP-SwRS-0465]

VIOM1VitalOut，VIOM2VitalOut，ATP输出给VIOM的车辆安全控制命令。

* 对于每一个端口的具体含义，是由项目配置的。ATP支持的可配置端口如Table 5‑14所示。
* 只有当CC未工作超时且CCNV未禁止该端口输出时，才能根据ATP计算结果输出该端口；否则，ATP默认该端口为限制状态。

def [VIOM1VitalOut](#VIOM1VitalOut)(k):

for port in [range](#range)(0, MAX\_VITAL\_OUTPUT\_NB):

if (not [CCworkOvertime](#CCworkOvertime)(k)

and not [MatchRebootCondition](#MatchRebootCondition)(k)

and [VIOM1OutNotDisabled](#VIOM1OutNotDisabled)(k)[port]):

[VIOM1VitalOut](#VIOM1VitalOut)[port] = [Offline](#Offline).[GetVIOM1VitalOut](#GetVIOM1VitalOut)(port)

else:

[VIOM1VitalOut](#VIOM1VitalOut)[port] = **False**

return [VIOM1VitalOut](#VIOM1VitalOut)

def [VIOM2VitalOut](#VIOM2VitalOut)(k):

for port in [range](#range)(0, MAX\_VITAL\_OUTPUT\_NB):

if (not [CCworkOvertime](#CCworkOvertime)(k)

and not [MatchRebootCondition](#MatchRebootCondition)(k)

and [VIOM2OutNotDisabled](#VIOM2OutNotDisabled)(k)[port]):

[VIOM2VitalOut](#VIOM2VitalOut)[port] = [Offline](#Offline).[GetVIOM2VitalOut](#GetVIOM2VitalOut)(port)

else:

[VIOM2VitalOut](#VIOM2VitalOut)[port] = **False**

return [VIOM2VitalOut](#VIOM2VitalOut)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0043], [iTC\_CC-SyAD-0114], [iTC\_CC-SyAD-0359], [iTC\_CC-SyAD-1129], [iTC\_CC-SyAD-0152], [iTC\_CC\_ATP\_SwHA-0017], [iTC\_CC\_ATP\_SwHA-0184], [iTC\_CC\_ATP\_SwHA-0270]

[End]

Table 5‑14 Configurable outputs for rolling stock

|  |  |  |
| --- | --- | --- |
| Name | | Description |
|  | [InhibitEmergencyBrake](#InhibitEmergencyBrake) | 是否禁止输出EB |
|  | [InhibitParkingBrake](#InhibitParkingBrake) | 是否禁止输出PB |
|  | [EnableTrainDoorOpening\_A](#EnableTrainDoorOpening_A) | 是否授权A侧开门 |
|  | [EnableTrainDoorOpening\_B](#EnableTrainDoorOpening_B) | 是否授权B侧开门 |
|  | [HoldDoorsClosed\_A](#HoldDoorsClosed_A) | 是否保持A侧门锁闭 |
|  | [HoldDoorsClosed\_B](#HoldDoorsClosed_B) | 是否保持B侧门锁闭 |
|  | [EmergencyDetrainDoorLockingEnd1](#EmergencyDetrainDoorLockingEnd1) | 是否保持End1端门锁闭 |
|  | [EmergencyDetrainDoorLockingEnd2](#EmergencyDetrainDoorLockingEnd2) | 是否保持End2端门锁闭 |
|  | [TrainFilteredStopped](#TrainFilteredStopped) | 是否ATP判断停车 |
|  | [TractionAuthorisedSenseEnd1](#TractionAuthorisedSenseEnd1) | 是否授权向End1端牵引 |
|  | [TractionAuthorisedSenseEnd2](#TractionAuthorisedSenseEnd2) | 是否授权向End2端牵引 |
|  | [MotionProtectionInhibition](#MotionProtectionInhibition) | 当前是否处于ATP监控模式 |

[iTC\_CC\_ATP-SwRS-0455]

根据[REF5]，在ATP发送给VIOM的命令中还应附加当前时间[ATPtime](#ATPtime)，上下CPU模块的周期同步校核字[Trace](#Trace)和[Dt](#Dt)，安全时钟状态[SafeTimerFailed](#SafeTimerFailed)，是否主控CC信息[MasterCCcore](#MasterCCcore)，以及需要反馈给各自VIOM的时间信息[LatestVIOM1LoopHourVIOM](#LatestVIOM1LoopHourVIOM)[2]和[LatestVIOM2LoopHourVIOM](#LatestVIOM2LoopHourVIOM)[2]。

According to [REF5], the commands ATP sent to VIOM shall attach the current ATPtime, cycle synchronization check word Trace and [Dt](#Dt), safety clock state [SafeTimerFailed](#SafeTimerFailed), [MasterCCcore](#MasterCCcore), the [LatestVIOM1LoopHourVIOM](#LatestVIOM1LoopHourVIOM) and [LatestVIOM2LoopHourVIOM](#LatestVIOM2LoopHourVIOM) needed to feedback to the respective VIOM.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0114], [iTC\_CC\_ATP\_SwHA-0017], [iTC\_CC\_ATP\_SwHA-0271]

[End]

[iTC\_CC\_ATP-SwRS-0752]

OutOfCode，ATP计算输出给VIOM的消息后，需要进行VIOM消息的VCP签名检测，用于维护诊断。

* 如果检测出签名有误，或上周期该值已经为**True**，则保持设置[OutOfCode](#OutOfCode)为**True**，表明ATP系统错误，此时需在VLE前面板LED显示**ERR\_OUT\_CODE**信息；
* 否则，设置[OutOfCode](#OutOfCode)为**False**。

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0408], [iTC\_CC-SyAD-0114], [iTC\_CC-SyAD-0386]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [InhibitEmergencyBrake](#InhibitEmergencyBrake) | √ | √ | BOOLEAN |
| [InhibitParkingBrake](#InhibitParkingBrake) | √ | √ | BOOLEAN |
| [OutOfCode](#OutOfCode) | √ | √ | BOOLEAN |
| [VIOM1VitalOut](#VIOM1VitalOut) | √ | √ | BOOLEAN |
| [VIOM2VItalOut](#VIOM2VItalOut) | √ | √ | BOOLEAN |

## F72-Outputs to ZC

ATP应当尽可能向ZC发送位置报告，即上电后一旦与LC初始化成功，并且获取到所管辖当前线路的ZC的编号时，就与该ZC进行通信。当列车车身跨2个ZC时，ATP应当同时向这2个ZC发送位置报告。位置报告中包括列车的识别信息，ATP工作状态，以及列车定位情况等，供ZC计算本列车的AP范围、属性以及EOA。

ATP shall send the Location Report to [ZC](#ZC) as far as possible. After ATP powered up and the communication with LC had fulfilled, and get the current [ZC](#ZC) id, it shall communicate with [ZC](#ZC). When the train passes across two [ZC](#ZC), ATP should also send the Location Report to both [ZC](#ZC). The information of Location Report includes train recognition information, ATP status, and train location, so that [ZC](#ZC) can calculate the AP scope, properties and EOA of the train.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATOcontrolTimeValid](#ATOcontrolTimeValid) | Internal | F11-Acquire Configuration Data |
| [ATPsetting](#ATPsetting) | Internal | F11-Acquire Configuration Data |
| [BeaconLastObtained](#BeaconLastObtained) | Internal | F27-Compute Train Kinematics |
| [DateSynchronizationReport](#DateSynchronizationReport) | External | Interface with LC |
| [EnableDoorOpening\_A](#EnableDoorOpening_A) | Internal | F61-Elaborate Door Opening Authorization |
| [EnableDoorOpening\_B](#EnableDoorOpening_B) | Internal | F61-Elaborate Door Opening Authorization |
| [LocationNotUncertaintyExceed](#LocationNotUncertaintyExceed) | Internal | F33-Confirm Train Localization |
| [MasterCCcore](#MasterCCcore) | Internal | F62-Control PSD Opening and Closing Order |
| [MotionProtectionInhibition](#MotionProtectionInhibition) | Internal | F46-Determine Over Energy |
| [NewBeaconObtained](#NewBeaconObtained) | Internal | F27-Compute Train Kinematics |
| [NonVitalRequest](#NonVitalRequest) | External | Interface with CC Non Vital |
| [Offline](#Offline) | Internal | F11-Acquire Configuration Data |
| [RMRselectedDrivingMode](#RMRselectedDrivingMode) | Internal | F54-Monitor Rollback Train Speed |
| [TrainCoupledType](#TrainCoupledType) | Internal | F12-Manage Train Status |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFilteredStopped](#TrainFilteredStopped) | Internal | F27-Compute Train Kinematics |
| [TrainFrontEnd](#TrainFrontEnd) | Internal | F33-Confirm Train Localization |
| [TrainFrontLocation](#TrainFrontLocation) | Internal | F33-Confirm Train Localization |
| [TrainLocalized](#TrainLocalized) | Internal | F32-Update Train Location |
| [TrainLocatedOnKnownPath](#TrainLocatedOnKnownPath) | Internal | F33-Confirm Train Localization |
| [TrainLocation](#TrainLocation) | Internal | F32-Update Train Location |
| [TrainMaxSpeed](#TrainMaxSpeed) | Internal | F27-Compute Train Kinematics |
| [TrainRearLocation](#TrainRearLocation) | Internal | F33-Confirm Train Localization |
| [VersionAuthorizedByLC](#VersionAuthorizedByLC) | Internal | F15-Manage Variants in CBTC Mode |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [DateSynchronisationReportAvailable](#DateSynchronisationReportAvailable) | √ | √ | BOOLEAN |
| [DateSynchronizationReceived](#DateSynchronizationReceived) | √ | √ | BOOLEAN |
| [EnableSendLocReport](#EnableSendLocReport) | √ | √ | BOOLEAN |
| [GroundTimeReference](#GroundTimeReference) | √ | √ | ST\_GROUND\_TIME |
| [LCsynchronisationLoopHourValid](#LCsynchronisationLoopHourValid) | √ | √ | BOOLEAN |
| [LastSynchronisationReportAge](#LastSynchronisationReportAge) | √ | √ | NUMERIC\_32 |
| [NonVitalRouteSetNotNeeded](#NonVitalRouteSetNotNeeded) | √ | √ | BOOLEAN |
| [SendLocReportOnZCunderTrainHead](#SendLocReportOnZCunderTrainHead) | √ | √ | BOOLEAN |
| [SendLocReportOnZCunderTrainTail](#SendLocReportOnZCunderTrainTail) | √ | √ | BOOLEAN |
| [SignalOverrideSendable](#SignalOverrideSendable) | √ | √ | BOOLEAN |
| [TimeElapseBetweenTwoLocReport](#TimeElapseBetweenTwoLocReport) | √ | √ | NUMERIC\_32 |
| [ZCidUnderTrainHead](#ZCidUnderTrainHead) | √ | √ | NUMERIC\_32 |
| [ZCidUnderTrainTail](#ZCidUnderTrainTail) | √ | √ | NUMERIC\_32 |

### Processing

#### Location Report creation

[iTC\_CC\_ATP-SwRS-0401]

TrainHeadOrientation，ATP需将车头最小定位的运营方向作为列车运营方向发送给ZC。规则见SwRS-0403。

ATP shall send the orientation of minimum location of current active cab id to the [ZC](#ZC). For the rule can refer to SwRS-0403.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0211]

[End]

[iTC\_CC\_ATP-SwRS-0403]

TrainHeadMinLocation，车头最小定位位置。

根据[REF5]，在[LocReport](#LocReport)中的坐标单位为0.5米，因此需进行单位转换。转换时应当向列车的“后方”即上游方向取整。

* 如果列车失位，则设置相关定位信息为无效值；
* 否则，如果列车向**UP**方向运行，则：
* 车头最小定位所在BLOCK号不变；
* 车头最小定位所在坐标以0.5米为单位向下取整；
* 车头方向为**LOCREPORT\_DIRECTION\_UP**。
* 否则，如果车头最小定位坐标加0.5米小于该BLOCK长度，则：
* 车头最小定位所在BLOCK号不变；
* 车头最小定位所在坐标以0.5米为单位向上取整；
* 车头方向为**LOCREPORT\_DIRECTION\_DOWN**。
* 否则，如果车头最小定位所在BLOCK，与该BLOCK的**UP**方向下个BLOCK之间存在灯泡线极点，则：
* 车头最小定位所在BLOCK需改为其**UP**方向的下个BLOCK；
* 车头最小定位所在坐标为下个BLOCK长度以0.5米为单位向下取整；
* 车头方向为**LOCREPORT\_DIRECTION\_UP**。
* 否则：
* 车头最小定位所在BLOCK需改为其**UP**方向的下个BLOCK；
* 车头最小定位所在坐标为0；
* 车头方向为**LOCREPORT\_DIRECTION\_DOWN**。

ATP shall send the minimum head location of the active cab to the [ZC](#ZC), including the block id and its abscissa. According to [REF5], the unit of the abscissa in Location Report is 0.5 meter, so the ATP needs to convert its internal unit to match that. The conversion shall be safety-oriented, which means the envelope of the train location tend to be "stretched" to the both ends. The rules of conversion are as following ARDL:

def [TrainHeadMinLocation](#TrainHeadMinLocation)(k):

if (not [TrainLocalized](#TrainLocalized)(k)):

[TrainHeadMinLocation](#TrainHeadMinLocation).Block = 0

[TrainHeadMinLocation](#TrainHeadMinLocation).Abscissa= 0

[TrainHeadOrientation](#TrainHeadOrientation) = **LOCREPORT\_DIRECTION\_UNKNOWN**

elif ([TrainFrontLocation](#TrainFrontLocation)(k).Min.Ort is **UP**):

[TrainHeadMinLocation](#TrainHeadMinLocation).Block = [TrainFrontLocation](#TrainFrontLocation)(k).Min.Block(k)

[TrainHeadMinLocation](#TrainHeadMinLocation).Abscissa = ([round.floor](#roundfloor)([TrainFrontLocation](#TrainFrontLocation)(k).Min.Abscissa(k)

/ **ABSCISSA\_TO\_HALF\_METER**))

[TrainHeadOrientation](#TrainHeadOrientation) = **LOCREPORT\_DIRECTION\_UP**

elif ([TrainFrontLocation](#TrainFrontLocation)(k).Min.Abscissa(k) + **ABSCISSA\_TO\_HALF\_METER**

<= [TrackMap](#TrackMap).[Blocks](#Blocks)[[TrainFrontLocation](#TrainFrontLocation)(k).Min.Block].Length):

[TrainHeadMinLocation](#TrainHeadMinLocation).Block = [TrainFrontLocation](#TrainFrontLocation)(k).Min.Block(k)

[TrainHeadMinLocation](#TrainHeadMinLocation).Abscissa = [round.ceil](#roundceil)([TrainFrontLocation](#TrainFrontLocation)(k).Min.Abscissa(k)

/ **ABSCISSA\_TO\_HALF\_METER**)

[TrainHeadOrientation](#TrainHeadOrientation) = **LOCREPORT\_DIRECTION\_DOWN**

else:

NextBlock = [TrackMap](#TrackMap).[NextBlock](#NextBlock)([TrackMap](#TrackMap).[Blocks](#Blocks)[[TrainFrontLocation](#TrainFrontLocation)(k).Min.Block], **UP**)

if [TrackMap.ExistThePole](#ExistThePole)([TrainFrontLocation](#TrainFrontLocation)(k).Min.Block, NextBlock.Id):

[TrainHeadMinLocation](#TrainHeadMinLocation).Block = NextBlock.Id

[TrainHeadMinLocation](#TrainHeadMinLocation).Abscissa = [round.floor](#roundfloor)(NextBlock.Length

/ **ABSCISSA\_TO\_HALF\_METER**)

[TrainHeadOrientation](#TrainHeadOrientation) = **LOCREPORT\_DIRECTION\_UP**

else:

[TrainHeadMinLocation](#TrainHeadMinLocation).Block = NextBlock.Id

[TrainHeadMinLocation](#TrainHeadMinLocation).Abscissa = 0

[TrainHeadOrientation](#TrainHeadOrientation) = **LOCREPORT\_DIRECTION\_DOWN**

return [TrainHeadMinLocation](#TrainHeadMinLocation)

In above ARDL, the **ABSCISSA\_TO\_HALF\_METER** means the coefficient of unit conversion.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0211], [iTC\_CC\_ATP\_SwHA-0196]

[End]

[iTC\_CC\_ATP-SwRS-0404]

TrainHeadCoupledStatus，车头连挂状态。

ATP shall send the coupled status of the active train cab to the [ZC](#ZC).

def [TrainHeadCoupledStatus](#TrainHeadCoupledStatus)(k):

if ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**):

return (([TrainCoupledType](#TrainCoupledType)(k) is **TRAIN\_NOT\_COUPLED**)

or ([TrainCoupledType](#TrainCoupledType)(k) is **TRAIN\_COUPLED\_END2**))

else:

return (([TrainCoupledType](#TrainCoupledType)(k) is **TRAIN\_NOT\_COUPLED**)

or ([TrainCoupledType](#TrainCoupledType)(k) is **TRAIN\_COUPLED\_END1**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0211]

[End]

[iTC\_CC\_ATP-SwRS-0405]

TrainTailCabId，车尾ID号。

ATP shall send the current inactive cab id to the [ZC](#ZC).

def [TrainTailCabId](#TrainTailCabId)(k):

if ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**):

return **END\_2**

else:

return **END\_1**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0211]

[End]

[iTC\_CC\_ATP-SwRS-0406]

TrainTailOrientation，ATP需将车尾最小定位的运营方向发送给ZC，作为车位运营方向。规则见SwRS-0408。

ATP shall send the orientation of the minimum location of inactive cab id to the [ZC](#ZC). For the rule can refer to SwRS-0408.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0211]

[End]

[iTC\_CC\_ATP-SwRS-0408]

TrainTailMinLocation，车尾最小定位所在位置。

根据[REF5]，在[LocReport](#LocReport)中的坐标单位为0.5米，因此需进行单位转换。转换时应当导向ZC处理的安全侧，偏向将列车包络定位“拉长”，即向列车的上游方向取整。

* 如果列车失位，设置上述定位信息为0；
* 否则，如果列车车尾朝**DOWN**方向，则：
* 车尾最小定位所在BLOCK号不变；
* 车尾最小定位所在坐标以0.5米为单位向下取整；
* 车尾方向为**LOCREPORT\_DIRECTION\_DOWN**。
* 否则，如果车尾最小定位坐标加0.5米小于该BLOCK长度，则：
* 车尾最小定位所在BLOCK号不变；
* 车尾最小定位所在坐标以0.5米为单位向上取整；
* 车尾方向为**LOCREPORT\_DIRECTION\_UP**。
* 否则，如果车尾最小定位所在BLOCK找不到**UP**方向的下个BLOCK，则：
* 车尾最小定位所在BLOCK号不变，
* 车尾最小定位所在坐标以0.5米为单位向上取整（此时由于发送的坐标超过了Block长度，ZC会将本LocReport丢弃，不会影响安全）。
* 车尾方向为**LOCREPORT\_DIRECTION\_UP**。
* 否则，如果车尾最小定位所在BLOCK，与该BLOCK的**UP**方向下个BLOCK之间存在灯泡线极点，则：
* 车尾最小定位所在BLOCK需改为其上行方向的下个BLOCK；
* 车尾最小定位所在坐标为下游BLOCK长度以0.5米为单位向下取整；
* 车尾方向为**LOCREPORT\_DIRECTION\_DOWN**。
* 否则，
* 车尾最小定位所在BLOCK需改为其上行方向的下个BLOCK；
* 车尾最小定位所在坐标为0；
* 车尾方向为**LOCREPORT\_DIRECTION\_UP**。

ATP shall send the minimum tail location of the active cab to the [ZC](#ZC), including the block id and its abscissa. According to [REF5], the unit of the abscissa in Location Report is 0.5 meter, so the ATP needs to convert its internal unit to match that. The convertion shall be safety-oriented, which means the envelope of the train location tend to be "stretched" to the both ends. The rules of convertion are as following ARDL:

def [TrainTailMinLocation](#TrainTailMinLocation)(k):

if (not [TrainLocalized](#TrainLocalized)(k)):

[TrainTailMinLocation](#TrainTailMinLocation).Block = 0

[TrainTailMinLocation](#TrainTailMinLocation).Abscissa = 0

[TrainTailOrientation](#TrainTailOrientation) = **LOCREPORT\_DIRECTION\_UNKOWN**

elif ([TrainRearLocation](#TrainRearLocation)(k).Min.Ort is **DOWN**):

[TrainTailMinLocation](#TrainTailMinLocation).Block = [TrainRearLocation](#TrainRearLocation)(k).Min.Block

[TrainTailMinLocation](#TrainTailMinLocation).Abscissa = ([round.floor](#roundfloor)([TrainRearLocation](#TrainRearLocation)(k).Min.Abscissa

/ **ABSCISSA\_TO\_HALF\_METER**))

[TrainTailOrientation](#TrainTailOrientation) = **LOCREPORT\_DIRECTION\_DOWN**

elif ([TrainRearLocation](#TrainRearLocation)(k).Min.Abscissa + **ABSCISSA\_TO\_HALF\_METER**

<= [TrackMap](#TrackMap).Block[[TrainRearLocation](#TrainRearLocation)(k).Min.Block].Length):

[TrainTailMinLocation](#TrainTailMinLocation).Block = [TrainRearLocation](#TrainRearLocation)(k).Min.Block

[TrainTailMinLocation](#TrainTailMinLocation).Abscissa = [round.ceil](#roundceil)([TrainRearLocation](#TrainRearLocation)(k).Min.Abscissa

/ **ABSCISSA\_TO\_HALF\_METER**)

[TrainTailOrientation](#TrainTailOrientation) = **LOCREPORT\_DIRECTION\_UP**

else:

NextBlock = [TrackMap](#TrackMap).[NextBlock](#NextBlock)([TrackMap](#TrackMap).[Blocks](#Blocks)[[TrainRearLocation](#TrainRearLocation)(k).Min.Block], **UP**)

if (NextBlock is **None**):

[TrainTailMinLocation](#TrainTailMinLocation).Block = [TrainFrontLocation](#TrainFrontLocation)(k).Min.Block

[TrainTailMinLocation](#TrainTailMinLocation).Abscissa = [round.ceil](#roundfloor)([TrainRearLocation](#TrainRearLocation)(k).Min.Abscissa

/ **ABSCISSA\_TO\_HALF\_METER**)

[TrainTailOrientation](#TrainTailOrientation) = **LOCREPORT\_DIRECTION\_UP**

elif ([TrackMap.ExistThePole](#ExistThePole)([TrainRearLocation](#TrainRearLocation)(k).Min.Block, NextBlock.Id)):

[TrainTailMinLocation](#TrainTailMinLocation).Block = NextBlock.Id

[TrainTailMinLocation](#TrainTailMinLocation).Abscissa = [round.floor](#roundfloor)(NextBlock.Length

/ **ABSCISSA\_TO\_HALF\_METER**)

[TrainTailOrientation](#TrainTailOrientation) = **LOCREPORT\_DIRECTION\_DOWN**

else:

[TrainTailMinLocation](#TrainTailMinLocation).Block = NextBlock.Id

[TrainTailMinLocation](#TrainTailMinLocation).Abscissa = 0

[TrainTailOrientation](#TrainTailOrientation) = **LOCREPORT\_DIRECTION\_UP**

return [TrainTailMinLocation](#TrainTailMinLocation)

In above ARDL, the **ABSCISSA\_TO\_HALF\_METER** means the coefficient of unit convertion.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0211], [iTC\_CC\_ATP\_SwHA-0196]

[End]

[iTC\_CC\_ATP-SwRS-0409]

TrainTailCoupledStatus，车尾连挂状态。

ATP shall send the coupled status of the inactive train cab to the [ZC](#ZC).

def [TrainTailCoupledStatus](#TrainTailCoupledStatus)(k):

if ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**):

return (([TrainCoupledType](#TrainCoupledType)(k) is **TRAIN\_NOT\_COUPLED**)

or ([TrainCoupledType](#TrainCoupledType)(k) is **TRAIN\_COUPLED\_END1**))

else:

return (([TrainCoupledType](#TrainCoupledType)(k) is **TRAIN\_NOT\_COUPLED**)

or ([TrainCoupledType](#TrainCoupledType)(k) is **TRAIN\_COUPLED\_END2**))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0211]

[End]

[iTC\_CC\_ATP-SwRS-0410]

LocationError，最大最小定位误差. 根据[REF5]，在LocReport中的长度单位是0.5米，因此需进行转换，转换时应当导向安全侧。

ATP shall send the location error to the [ZC](#ZC). According to [REF5], the unit of the location error in Location Report is 0.5 meter, so the ATP needs to convert its internal unit to match that. The convertion shall be safety-oriented, which means the location error tend to be "overestimated".

def LocationError(k):

if [TrainLocalized](#TrainLocalized)(k):

return [round.ceil](#roundceil)(([TrainLocation](#TrainLocation)(k).Uncertainty + **ABSCISSA\_TO\_HALF\_METER**)

/ **ABSCISSA\_TO\_HALF\_METER**)

else:

return 0

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0212], [iTC\_CC\_ATP\_SwHA-0196]

[End]

[iTC\_CC\_ATP-SwRS-0068]

RouteSetNotNeededSendable，是否可以发送RSNN信息。其状态来自于项目可配置的列车输入采集。

According to the status of [RouteSetNotNeededSendable](#RouteSetNotNeededSendable), ATP can judge whether it is necessary to send RSNN information.

def [RouteSetNotNeededSendable](#RouteSetNotNeededSendable)(k):

return [Offline.GetRouteSetNotNeededSendable](#GetRouteSetNotNeededSendable)()

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0215], [iTC\_CC-SyAD-0237], [iTC\_CC-SyAD-0341], [iTC\_CC-SyAD-1003], [iTC\_CC-SyAD-1311], [iTC\_CC\_ATP\_SwHA-0204]

[End]

[iTC\_CC\_ATP-SwRS-0135]

NonVitalRouteSetNotNeeded，RSNN状态

Whether the CCNV request route set note needed.

def [NonVitalRouteSetNotNeeded](#NonVitalRouteSetNotNeeded)(k):

return ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k)

and [NonVitalRequest](#NonVitalRequest).RouteSetNotNeeded(k))

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0215], [iTC\_CC-SyAD-0408], [iTC\_CC-SyAD-1044]

[End]

[iTC\_CC\_ATP-SwRS-0414]

TrainRouteSetNotNeeded，是否发送RSNN信息。

ATP shall send the route set not needed information to [ZC](#ZC).

def [TrainRouteSetNotNeeded](#TrainRouteSetNotNeeded)(k):

return ([TrainFilteredStopped](#TrainFilteredStopped)(k)

and [NonVitalRouteSetNotNeeded](#NonVitalRouteSetNotNeeded)(k)

and [RouteSetNotNeededSendable](#RouteSetNotNeededSendable)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0215], [iTC\_CC-SyAD-0237], [iTC\_CC\_ATP\_SwHA-0210]

[End]

[iTC\_CC\_ATP-SwRS-0415]

TrainCorrectDocking，列车是否正确停靠车站。

ATP shall send the docking correction information to the [ZC](#ZC).

def [TrainCorrectDocking](#TrainCorrectDocking)(k):

return ([EnableDoorOpening\_A](#EnableDoorOpening_A)(k) or [EnableDoorOpening\_B](#EnableDoorOpening_B)(k))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0216]

[End]

[iTC\_CC\_ATP-SwRS-0416]

LocReportSpeed，列车最大速度，需转换为CC-ZC接口协议中的单位，并向上取整。

ATP shall send the maximum train speed to the [ZC](#ZC). According to [REF5], the unit of the speed in Location Report is KPH, so the ATP needs to convert its internal unit to match that. The convertion shall be safety-oriented, which means the speed tend to be "overestimated".

def [LocReportSpeed](#LocReportSpeed)(k):

return [round.ceil](#roundceil)([TrainMaxSpeed](#TrainMaxSpeed)(k) / **KMPH\_TO\_MMPS**)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0217], [iTC\_CC\_ATP\_SwHA-0272]

[End]

[iTC\_CC\_ATP-SwRS-0417]

TrainMonitoringMode，监控模式.

ATP shall send the current monitoring mode to the [ZC](#ZC).

def [TrainMonitoringMode](#TrainMonitoringMode)(k):

if ([MotionProtectionInhibition](#MotionProtectionInhibition)(k)

and [RMRselectedDrivingMode](#RMRselectedDrivingMode)(k)):

return **RMR**

elif ([MotionProtectionInhibition](#MotionProtectionInhibition)(k)):

return **RMF**

else:

return **OTHERS**

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0218], [iTC\_CC-SyAD-0344], [iTC\_CC\_ATP\_SwHA-0212]

[End]

[iTC\_CC\_ATP-SwRS-0599]

SignalOverrideSendable，发给ZC的关信号机命令。

def [SignalOverrideSendable](#SignalOverrideSendable)(k):

return [Offline.GetSignalOverrideSendable](#GetSignalOverrideSendable)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1312]

[End]

[iTC\_CC\_ATP-SwRS-0418]

SignalsOverride，是否CBTC下取消信号。

ATP shall send the signal override information to the [ZC](#ZC).

def [SignalsOverride](#SignalsOverride)(k):

return ([SignalOverrideSendable](#SignalOverrideSendable)(k)

and not [MotionProtectionInhibition](#MotionProtectionInhibition)(k)

and ([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k)

and [NonVitalRequest](#NonVitalRequest).CancelSignal))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-0219], [iTC\_CC\_ATP\_SwHA-0221]

[End]

[iTC\_CC\_ATP-SwRS-0598]

ATCcontrolledTrain，ATP未被切除。

def [ATCcontrolledTrain](#ATCcontrolledTrain)(k):

return [Offline.GetATCcontrolledTrain](#GetATCcontrolledTrain)(k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1216], [iTC\_CC-SyAD-1296], [iTC\_CC-SyAD-1297], [iTC\_CC-SyAD-1306]

[End]

#### Track side time

[iTC\_CC\_ATP-SwRS-0728]

DateSynchronizationReceived，收到时钟同步消息

def [DateSynchronizationReceived](#DateSynchronizationReceived)(lcId, k):

return [Message.Received](#Received)([DateSynchronizationReport](#DateSynchronizationReport)(lcId), k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0383]

[End]

[iTC\_CC\_ATP-SwRS-0094]

DateSynchronisationReportAvailable用于判断当前周期收到的LC安全时间消息是否可用。当满足以下条件时，设置DateSynchronisationReportAvailable为**True**。否则，设置DateSynchronisationReportAvailable为**False**。

* 本周期收到来自LC的[DateSynchronizationReport](#DateSynchronizationReport)消息，并且校核字正确；
* 如果收到LC消息是LC应答本端CC发出的消息，且满足以下条件时:
* 当前ATP时间应大于消息中所带的ccLoopHour;
* 并且当前ATP时间与消息中所带的ccLoopHour的差值应当小于之前ATP使用的LC消息时间。
* 如果该消息是LC应答远端CC发出的消息，则应满足以下条件:
* 当前ATP维护的远端ATP最小时间应大于消息中所带的ccLoopHour;
* 并且当前ATP维护的远端ATP最大时间时间与消息中所带的ccLoopHour的差值，应当小于之前ATP使用的LC消息时间。

[DateSynchronisationReportAvailable](#DateSynchronisationReportAvailable) used to judge whether the LC synchronization message could use or not. When the below conditions fulfilled, [DateSynchronisationReportAvailable](#DateSynchronisationReportAvailable) shall set as **True**. Otherwise, it set as **False**.

* If ATP Receives the [DateSynchronizationReport](#DateSynchronizationReport) message from LC, and the checksum is correct.
* And If the received message is LC to respond the local CC, ATP shall qualify with below conditions:
* The current ATP time is more than ccLoopHour in the message
* The different value between the current ATP time and the ccLoopHour is less than the LC message time in the previous ATP.
* Or else:, If the received message is LC to respond the remote CC, and it should be qualified with below conditions:
* The minimum time in remote ATP maintained by current ATP is more than the ccLoopHour
* The different value between maximum time in remote ATP maintained by current ATP and the ccLoopHour is less than the LC message time in the previous ATP.

def [DateSynchronisationReportAvailable](#DateSynchronisationReportAvailable)(lcId, k):

return Message.[Available](#Available)([DateSynchronizationReceived](#DateSynchronizationReceived)(lcId, k),

[DateSynchronizationReport](#DateSynchronizationReport)(lcId).CcLoopHour,

[ATPsetting](#ATPsetting).LCloophourValidityTime,

[LastSynchronisationReportAge](#LastSynchronisationReportAge)(lcId, k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0383], [iTC\_CC\_ATP\_SwHA-0019]

[End]

[iTC\_CC\_ATP-SwRS-0095]

LastSynchronisationReportAge，记录从上次收到LC消息到现在的时间。用于在新收到LC消息时，比较该“新”消息是否的确比之前的消息“新”。即防止在网络传输中发生消息逆序的情况。

[LastSynchronisationReportAge](#LastSynchronisationReportAge) records the age from previous LC message to current cycle. It used to compare whether the new received message is more updating than last recorded one, to prevent the inverse transition in the network communication.

def [LastSynchronisationReportAge](#LastSynchronisationReportAge)(lcId, k):

return Message.[LastAge](#LastAge)([DateSynchronisationReportAvailable](#DateSynchronisationReportAvailable)(lcId, k),

[DateSynchronizationReport](#DateSynchronizationReport)(lcId).CcLoopHour,

[LastSynchronisationReportAge](#LastSynchronisationReportAge)(lcId, k-1),

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0383]

[End]

[iTC\_CC\_ATP-SwRS-0097]

LCsynchronisationLoopHourValid，每周期更新LC消息是否还在有效期内。

[LCsynchronisationLoopHourValid](#LCsynchronisationLoopHourValid) will check whether the LC message is still valid.

def [LCsynchronisationLoopHourValid](#LCsynchronisationLoopHourValid)(lcId, k):

return Message.[Valid](#Valid)([DateSynchronizationReport](#DateSynchronizationReport)(lcId).CcLoopHour,

[ATPsetting](#ATPsetting).LCloophourValidityTime,

k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0384], [iTC\_CC\_ATP\_SwHA-0274]

[End]

[iTC\_CC\_ATP-SwRS-0098]

对于线路上的LC，ATP需要维护其时间信息GroundTimeReference，结构为ST\_GROUND\_TIME：

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_GROUND\_TIME | |  |  |
|  | Time | NUMERIC\_32 | 以轨旁周期号为单位的时间 |
|  | Milliseconds | NUMERIC\_32 | 以毫秒为单位累加轨旁时间 |

时间信息更新规则如下：

* LC消息无效时，设为默认值；
* 本周期收到新的可用的LC消息时，更新为消息中的时间；
* 否则根据车载和轨旁周期的比值进行累加更新

ATP shall estimate a time called [GroundTimeReference](#GroundTimeReference) for trackside equipment ([ZC](#ZC) /LC). From power-up, [GroundTimeReference](#GroundTimeReference) shall consider invalid. The rules for time information are as follows:

* When LC message invalid, it is set as default value.
* When ATP receives an available LC message, it updates as the time of message.
* Otherwise, it updates the value according to the on board and track side cycle.

if (Initialization

or [LCsynchronisationLoopHourValid](#LCsynchronisationLoopHourValid)(LcId, k) != **True**)

[GroundTimeReference](#GroundTimeReference)[LcId].Time = **INVALID\_LC\_DATE**

[GroundTimeReference](#GroundTimeReference)[LcId].Milliseconds = 0

elif ([DateSynchronisationReportAvailable](#DateSynchronisationReportAvailable)(LcId, k))

[GroundTimeReference](#GroundTimeReference)[LcId].Time(k)= [DateSynchronizationReport](#DateSynchronizationReport).Synchrodate

[GroundTimeReference](#GroundTimeReference)[LcId].Milliseconds = 0

else:

[GroundTimeReference](#GroundTimeReference)[LcId].Time(k)

= [GroundTimeReference](#GroundTimeReference)[LcId].Time(k-1)

+ (([GroundTimeReference](#GroundTimeReference)[LcId].Milliseconds(k-1)+ **ATP\_CYCLE\_TIME\_MS**)

/ **SYNCHRODATE\_TIME\_UNIT\_MS**)

[GroundTimeReference](#GroundTimeReference)[LcId].Milliseconds(k)

= (([GroundTimeReference](#GroundTimeReference)[LcId].Milliseconds(k-1)+ **ATP\_CYCLE\_TIME\_MS**)

% **SYNCHRODATE\_TIME\_UNIT\_MS**)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0222], [iTC\_CC-SyAD-0381], [iTC\_CC-SyAD-0382], [iTC\_CC-SyAD-0383], [iTC\_CC-SyAD-1025], [iTC\_CC\_ATP\_SwHA-0020], [iTC\_CC\_ATP\_SwHA-0157], [iTC\_CC\_ATP\_SwHA-0274]

[End]

**NOTES:**

在ATP内部不使用[GroundTimeReference](#GroundTimeReference)参与计算，仅发送给ZC使用。对于每个ZC的[LocReport](#LocReport)，应当发送其所对应的LC的时间。

In ATP shall not use [GroundTimeReference](#GroundTimeReference) during the calculation of the internal functions, but only send the info to [ZC](#ZC).

#### Location Report sendable

[iTC\_CC\_ATP-SwRS-0437]

EnableSendLocReport，当所对应的LC消息在有效期内时，才允许给ZC发送位置报告。

Before received the first valid synchronization from a LC, the corresponding LC vital time shall consider not significant, and the ATP shall prevent to send Location Report message to the ZC(s) associated to the corresponding LC.

def [EnableSendLocReport](#EnableSendLocReport)(LcId, k):

return [LCsynchronisationLoopHourValid](#LCsynchronisationLoopHourValid)(LcId, k)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0206], [iTC\_CC-SyAD-0382], [iTC\_CC-SyAD-0384], [iTC\_CC-SyAD-0389], [iTC\_CC-SyAD-0913], [iTC\_CC\_ATP\_SwHA-0214]

[End]

[iTC\_CC\_ATP-SwRS-0438]

TimeElapseBetweenTwoLocReport，计算发送[LocReport](#LocReport)的时间控制：

ATP shall calculate when to send the Location Report as the following ARDL:

def [TimeElapseBetweenTwoLocReport](#TimeElapseBetweenTwoLocReport)(k):

if (Initialization

or ([TimeElapseBetweenTwoLocReport](#TimeElapseBetweenTwoLocReport)(k-1) == 1)):

return [round.floor](#floor)([ATPsetting](#ATPsetting).LocReportEmissionPeriod)

else:

return [TimeElapseBetweenTwoLocReport](#TimeElapseBetweenTwoLocReport)(k-1) - 1

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0207], [iTC\_CC-SyAD-1004]

[End]

[iTC\_CC\_ATP-SwRS-0439]

ZCidUnderTrainTail，车尾定位所在的ZC标识.

ATP shall record the [ZC](#ZC) area id where the train tail located. If the train is not localized, ATP shall use the current receiving beacon where located, or the last known [ZC](#ZC) area id.

def [ZCidUnderTrainTail](#ZCidUnderTrainTail)(k):

if (Initialization):

return 0

elif ([TrainLocalized](#TrainLocalized)(k)):

return [TrackMap](#TrackMap).[ZCId](#ZCId)([TrainTailMinLocation](#TrainTailMinLocation)(k).Block)

elif ([MemorizedLocationAvailable](#MemorizedLocationAvailable)(k)):

if ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**):

return [TrackMap](#TrackMap).[ZCId](#ZCId)([MemLocation](#MemLocation).Ext2.Block)

else:

return [TrackMap](#TrackMap).[ZCId](#ZCId)([MemLocation](#MemLocation).Ext1.Block)

elif ([NewBeaconObtained](#NewBeaconObtained)(k)):

return [TrackMap](#TrackMap).[ZCId](#ZCId)([BeaconLastObtained](#BeaconLastObtained).Block)

else:

return [ZCidUnderTrainTail](#ZCidUnderTrainTail)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0181], [iTC\_CC-SyAD-0205], [iTC\_CC-SyAD-0836], [iTC\_CC\_ATP\_SwHA-0197]

[End]

[iTC\_CC\_ATP-SwRS-0440]

ZCidUnderTrainHead，车头定位的ZC标识

ATP shall record the [ZC](#ZC) area id where the train head located. If the train is not localized, ATP shall use the current receiving beacon where located, or the last known [ZC](#ZC) area id.

def [ZCidUnderTrainHead](#ZCidUnderTrainHead)(k):

if (Initialization)

return 0

elif ([TrainLocalized](#TrainLocalized)(k)):

return [TrackMap](#TrackMap).[ZCId](#ZCId)([TrainHeadMinLocation](#TrainHeadMinLocation)(k).Block)

elif ([MemorizedLocationAvailable](#MemorizedLocationAvailable)(k)):

if ([TrainFrontEnd](#TrainFrontEnd)(k) is **END\_1**):

return [TrackMap](#TrackMap).[ZCId](#ZCId)([MemLocation](#MemLocation).Ext1.Block)

else:

return [TrackMap](#TrackMap).[ZCId](#ZCId)([MemLocation](#MemLocation).Ext2.Block)

elif ([NewBeaconObtained](#NewBeaconObtained)(k)):

return [TrackMap](#TrackMap).[ZCId](#ZCId)([BeaconLastObtained](#BeaconLastObtained).Block)

else:

return [ZCidUnderTrainHead](#ZCidUnderTrainHead)(k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0181], [iTC\_CC-SyAD-0205], [iTC\_CC-SyAD-0836], [iTC\_CC\_ATP\_SwHA-0197]

[End]

[iTC\_CC\_ATP-SwRS-0441]

SendLocReportOnZCunderTrainTail，当距离上次发送[LocReport](#LocReport)到达既定时间，且车头和车尾不是同一个ZC时，要给车尾所在的ZC发送消息

When the time elapsed from last report is equal to the Location Report sending cycle, and the train tail located [ZC](#ZC) is valid and not as same as the train head [ZC](#ZC), and the [ZC](#ZC) train tail located authorized by the LC, ATP shall send the Location Report to this [ZC](#ZC) where train tail located.

def [SendLocReportOnZCunderTrainTail](#SendLocReportOnZCunderTrainTail)(k):

return ((([TimeElapseBetweenTwoLocReport](#TimeElapseBetweenTwoLocReport)(k) == 1)

or (not [MasterCCcore](#MasterCCcore)(k-1) and [MasterCCcore](#MasterCCcore)(k)))

and ([ZCidUnderTrainTail](#ZCidUnderTrainTail)(k) is not **None**)

and ([ZCidUnderTrainTail](#ZCidUnderTrainTail)(k) != [ZCidUnderTrainHead](#ZCidUnderTrainHead)(k))

and [EnableSendLocReport](#EnableSendLocReport)([TrackMap.ZC](#ZC)[[ZCidUnderTrainTail](#ZCidUnderTrainTail)(k)].LcId, k)

and ([VersionAuthorizedByLC](#VersionAuthorizedByLC)([ZCidUnderTrainTail](#ZCidUnderTrainTail)(k), k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0206], [iTC\_CC-SyAD-0389], [iTC\_CC\_ATP\_SwHA-0198]

[End]

[iTC\_CC\_ATP-SwRS-0442]

SendLocReportOnZCunderTrainHead，是否给车头的ZC区发[LocReport](#LocReport)。

When the time elapsed from last report is equal to the Location Report sending cycle, and the train head located [ZC](#ZC) is valid, and the [ZC](#ZC) train head located authorized by the LC, ATP shall send the Location Report to this [ZC](#ZC) where train head located.

def [SendLocReportOnZCunderTrainHead](#SendLocReportOnZCunderTrainHead)(k):

return ((([TimeElapseBetweenTwoLocReport](#TimeElapseBetweenTwoLocReport)(k) == 1)

or (not [MasterCCcore](#MasterCCcore)(k-1) and [MasterCCcore](#MasterCCcore)(k)))

and ([ZCidUnderTrainHead](#ZCidUnderTrainHead)(k) is not **None**)

and [EnableSendLocReport](#EnableSendLocReport)([TrackMap.ZC](#ZC)[[ZCidUnderTrainHead](#ZCidUnderTrainHead)(k)].LcId, k)

and ([VersionAuthorizedByLC](#VersionAuthorizedByLC)([ZCidUnderTrainHead](#ZCidUnderTrainHead)(k), k)))

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-0206], [iTC\_CC-SyAD-0389], [iTC\_CC\_ATP\_SwHA-0198]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [ATCcontrolledTrain](#ATCcontrolledTrain) | √ | √ | BOOLEAN |
| [LocReportSpeed](#LocReportSpeed) | √ | √ | NUMERIC\_32 |
| [LocationError](#LocationError) | √ | √ | BOOLEAN |
| [RouteSetNotNeededSendable](#RouteSetNotNeededSendable) | √ | √ | BOOLEAN |
| [SignalsOverride](#SignalsOverride) | √ | √ | BOOLEAN |
| [TrainCorrectDocking](#TrainCorrectDocking) | √ | √ | BOOLEAN |
| [TrainHeadCoupledStatus](#TrainHeadCoupledStatus) | √ | √ | BOOLEAN |
| [TrainheadMinLocation](#TrainheadMinLocation) | √ | √ | ST\_LOCATION\_UNIT |
| [TrainHeadOrientation](#TrainHeadOrientation) | √ | √ | ENUM\_DOT |
| [TrainMonitoringMode](#TrainMonitoringMode) | √ | √ | ENUM\_MONITORING\_MODE |
| [TrainRouteSetNotNeeded](#TrainRouteSetNotNeeded) | √ | √ | BOOLEAN |
| [TrainTailCabId](#TrainTailCabId) | √ | √ | NUMERIC\_32 |
| [TrainTailCoupledStatus](#TrainTailCoupledStatus) | √ | √ | BOOLEAN |
| [TrainTailMinLocation](#TrainTailMinLocation) | √ | √ | ST\_LOCATION\_UNIT |
| [TrainTailOrientation](#TrainTailOrientation) | √ | √ | ENUM\_DOT |

## F73-Outputs to CI Radio

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [CCvariantRequest](#CCvariantRequest) | External | Interface with CI Radio |
| [NonVitalRequest](#NonVitalRequest) | External | Interface with CC Non Vital |
| [TrackMap](#TrackMap) | Internal | F11-Acquire Configuration Data |
| [TrainFrontLocation](#TrainFrontLocation) | Internal | F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [CCvariantRequestMsgReceived](#CCvariantRequestMsgReceived) | √ | √ | BOOLEAN |

### Processing

在与联锁进行无线通信时，联锁会请求CC发送Overlap解锁信息。ATP应记录该请求，并在CCNV授权发送Overlap解锁信息时，根据车头最大定位所在的**SGL\_OVERLAP\_RELEASE\_ZONE**，发送解锁请求。

#### Receive CC variant request

[iTC\_CC\_ATP-SwRS-0729]

CCvariantRequestMsgReceived，收到来自CI的CC变量请求并校核字正确。

def [CCvariantRequestMsgReceived](#CCvariantRequestMsgReceived)(cbiId, k):

if (Initialization):

return **False**

elif ([Message.Received](#Received)([CCvariantRequest](#CCvariantRequest), k)):

return **True**

else:

return [CCvariantRequestMsgReceived](#CCvariantRequestMsgReceived)(Cbi, k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1193]

[End]

[iTC\_CC\_ATP-SwRS-0730]

ReceivedCBIloopHour，记录CBI请求中的联锁的loop hour。

def [ReceivedCBIloopHour](#ReceivedCBIloopHour)(cbi, k):

if (Initialization):

return **INVALID\_LOOP\_HOUR**

elif ([CCvariantRequestMsgReceived](#CCvariantRequestMsgReceived)(Cbi, k)):

return [CCvariantRequest](#CCvariantRequest)(Cbi, k).CbiLoopHour

else:

return [ReceivedCBIloopHour](#ReceivedCBIloopHour)(Cbi, k-1)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1195]

[End]

#### Send CC variant report

[iTC\_CC\_ATP-SwRS-0731]

CCvariants，ATP发送给联锁的Overlap解锁信息。

ATP shall check the following conditions when sending overlap release to CBI:

* Train front location is in overlap release zone,
* and the other ATP's overlap timer has expired,
* and ATP received variant request from CBI in this zone.

def [CCvariants](#CCvariants)([CbiId](#CbiId), k):

Orz = [TrackMap](#TrackMap).[ExistZoneLocationIncluded](#ExistZoneLocationIncluded)(**SGL\_OVERLAP\_RELEASE\_ZONE**,

[TrainFrontLocation](#TrainFrontLocation)(k).Max):

if ([OverlapReleasable](#OverlapReleasable)(k)

and [OtherATP](#OtherATP)(k).OverlapExpired

and [CCvariantRequestMsgReceived](#CCvariantRequestMsgReceived)(CbiId, k)

and Orz is not **None**

and Orz.[CbiId](#CbiId) == [CbiId](#CbiId)

and [CbiId](#CbiId) == [NonVitalRequest](#NonVitalRequest)(k).VariantRequestCbiId):

for Index in [range](#range)(0, MAX\_CC\_VARIANTS\_NB):

if (Orz.RadioBlockModeVariantIndex == Index):

[CCvariants](#CCvariants)[[CbiId](#CbiId)].Variant[Index] = **True**

else:

[CCvariants](#CCvariants)[[CbiId](#CbiId)].Variant[Index] = **False**

else:

[CCvariants](#CCvariants)[[CbiId](#CbiId)] = **None**

return [CCvariants](#CCvariants)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software, Vital Embedded Setting

#Source=[iTC\_CC-SyAD-1194], [iTC\_CC-SyAD-1195], [iTC\_CC-SyAD-1292], [iTC\_CC\_ATP\_SwHA-0273]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [CCvariants](#CCvariants) | √ | √ | BOOLEAN |
| [ReceivedCBIloopHour](#ReceivedCBIloopHour) | √ | √ | NUMERIC\_32 |

## F8-Platform Relative & Assist Functions

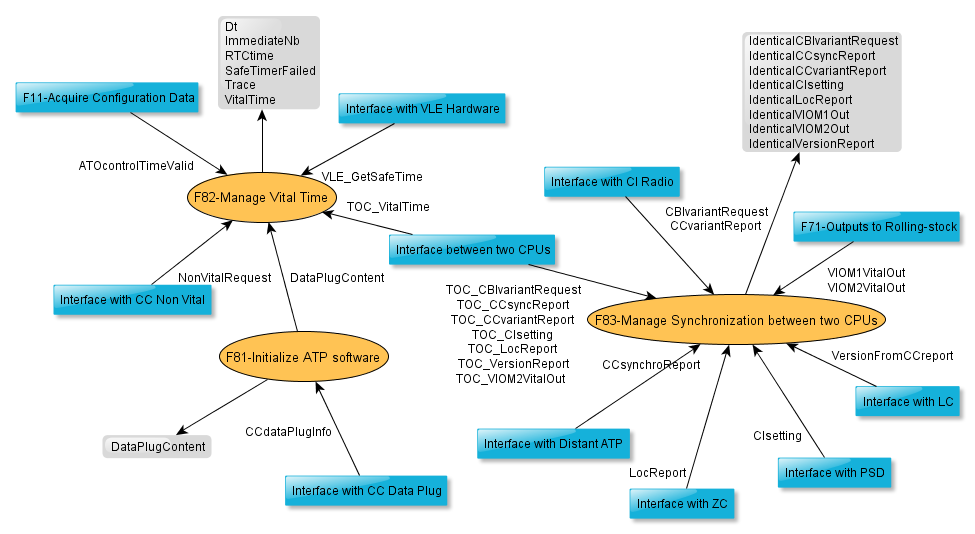


Figure 5‑22 SART modeling of function F8

## F81-Initialize ATP software

本模块用于控制ATP软件的初始化。

This function used to initialize the ATP software.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [CCdataPlugInfo](#CCdataPlugInfo) | External | Interface with CC Data Plug |

### Processing

车载系统上电后，ATP软件需执行下列初始化工作：

* 从data plug获取项目配置的车辆和CC基本识别信息；
* 初始化与CC内部其他系统的通信，如CCNV、DLU；
* 初始化VPB板。

#### Read data plug

[iTC\_CC\_ATP-SwRS-0511]

初始化时，ATP读取带VCP编码的来自CC data plug接口的信息[CCdataPlugInfo](#CCdataPlugInfo)，生成DataPlugContent，其结构如Table 4‑4所示。

In Initialization, ATP reads the [CCdataPlugInfo](#CCdataPlugInfo) with VCP coded from CC data plug, and generates [DataPlugContent](#DataPlugContent) with the structure shown as Table 4‑4。

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0067], [iTC\_CC\_ATP\_SwHA-0185]

[End]

[iTC\_CC\_ATP-SwRS-0037]

初始化时，ATP软件通过[DataPlugContent](#DataPlugContent).VLECpuId来识别所在的是CPU1还是CPU2，并与另一个CPU模块建立通信。

* 如果从Dataplug读到的cpuId错误，则导致双CPU建立通信失败，ATP软件禁止执行，等待操作人员手动重启系统；
* 对于CPU1，将在等待双CPU建立通信时控制LED显示**WAITING\_CPU**信息。
* 对于在CPU1运行的ATP软件，如果读到的cpuId既不是CPU1也不是CPU2时，控制LED显示**ERR\_CPU\_ID**信息。

Through [DataPlugContent](#DataPlugContent).VLECpuId, ATP recognizes whether it is located in CPU1 or CPU2, and establish the communication between each other.

* If the CPU id is wrong, ATP shall stop and wait for reboot manually by the operator.
* For CPU1, ATP shall control the LED to show **WAITING\_CPU** when the communication establishing.
* For CPU1, ATP shall control the LED to show **ERR\_CPU\_ID** when the cpuId is neither CPU1 nor CPU2.

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0831], [iTC\_CC-SyAD-0032], [iTC\_CC-SyAD-1423]

[End]

**NOTES**:

由于硬件设计限制，仅有CPU1的软件可以控制面板的LED显示。

Due to the limitation of hardware, only the software running on CPU1 can control the display of LED in the front panel of VLE-2 board.

#### CCNV initialization

[iTC\_CC\_ATP-SwRS-0056]

初始化时，ATP软件需将来自[DataPlugContent](#DataPlugContent)的部分内容，如Table 4‑7所示，在与CCNV建立通信后通过双口RAM发送给CCNV。如果ATP软件无法与CCNV建立通信，则保持等待，由CPU1在LED上显示**WAITING \_CCNV**，直到通信建立成功或者操作人员重启VLE-2板。

In initialization, ATP needs to send some part of contents as shown in Table 4‑7 from CC data plug to CCNV by dual-ports RAM after getting contact with CCNV. If ATP cannot establish the communication with CCNV, it will keep waiting and show in the LED as **WAITING \_CCNV**, until the communication is built or the operator reboots VLE-2 board.

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source= [iTC\_CC-SyAD-0067], [iTC\_CC-SyAD-0831], [iTC\_CC-SyAD-0032], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0034], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0071], [iTC\_CC\_VLE-2-DVCOM-2-SyID-0018], [iTC\_CC-SyAD-1423]

[End]

#### DLU initialization

[iTC\_CC\_ATP-SwRS-0512]

初始化时，ATP软件需将来自CC data plug的[DataPlugContent](#DataPlugContent).DLUIpBlue和DLUIpRed信息通过双口RAM发送给DLU。

In initialization, ATP needs to send the [DataPlugContent](#DataPlugContent). DLUIpBlue and DLUIpRed from CC data plug to DLU through RAM.

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0798], [iTC\_CC\_VLE-2-DLU-SyID-0004], [iTC\_CC\_VLE-2-DLU-SyID-0005], [iTC\_CC\_VLE-2-DLU-SyID-0006], [iTC\_CC\_VLE-2-DLU-SyID-0007]

[End]

#### VPB initialization

[iTC\_CC\_ATP-SwRS-0039]

初始化相关硬件，如VPB板，若硬件初始化成功则继续执行；若失败，则CPU1的ATP软件在LED上显示**ERR\_INIT\_VPB**信息，并禁止软件继续执行，等待操作人员手动重启系统。

The hardware needs to be initialized, e.g. VPB board. If the initialization of hardware succeeds, the system will continue. Otherwise, ATP on CPU1 shall show the message in the LED as **ERR\_INIT\_VPB**, and prohibit software executing and waiting for the system reboot by the operator.

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0064], [iTC\_CC-SyAD-0831], [iTC\_CC-SyAD-0032], [iTC\_CC-SyAD-1423]

[End]

[iTC\_CC\_ATP-SwRS-0591]

初始化时，ATP上下CPU模块应当比较读取自VPB板的[CBKWrite](#CBKWrite). SensorReg, CogCounterReg, CalibrationReg, StatusReg寄存器初值是否相同。若相同则继续执行，否则，应在LED上显示**UNCONST\_VPB**信息，并禁止软件继续执行。

In initialization, ATP in two CPU modules shall compare the values from VPB registers. If these initial values are equal, the ATP will continue; Otherwise, ATP shall show the message in LED as **UNCONST\_VPB**, and prohibit software executing.

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0032]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [DataPlugContent](#DataPlugContent) | √ | √ | ST\_DATA\_PLUG |

## F82-Manage Vital Time

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [ATOcontrolTimeValid](#ATOcontrolTimeValid) | Internal | F11-Acquire Configuration Data |
| [DataPlugContent](#DataPlugContent) | Internal | F81-Initialize ATP software |
| [NonVitalRequest](#NonVitalRequest) | External | Interface with CC Non Vital |
| [TOC\_VitalTime](#TOC_VitalTime) | External | Interface between two CPUs |
| [VLE\_GetSafeTime](#VLE_GetSafeTime) | External | Interface with VLE Hardware |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [CycleBiasNb](#CycleBiasNb) | √ | √ | NUMERIC\_32 |
| [[CycleSynchronized](#CycleSynchronized)](#CycleSynchroized) | √ | √ | BOOLEAN |
| [ImmediateCounter](#ImmediateCounter) | √ | √ | NUMERIC\_32 |
| [LockedImmediateCounter](#LockedImmediateCounter) | √ | √ | NUMERIC\_32 |
| [VLEimpulseNb](#VLEimpulseNb) | √ | × | NUMERIC\_32 |

### Processing

VLE-2板上的两个CPU模块各提供一个安全时钟，ATP软件在响应定时中断时，需获取该时钟的脉冲数，用于检测定时中断的正确性。此外，两个CPU上的ATP软件应当通过双口RAM进行周期同步，确保它们之间的主周期时间误差不超过一个定时中断的时长。从安全角度考虑，ATP周期时间的长度只能偏大，而不能偏小（因为ATP根据里程计的位移值除以预订的周期时间来计算速度，如果实际的周期时长比预计值偏大，则计算得到的速度也比实际值大，是导向安全的）。

Both CPU modules of the VLE-2 board have a safe clock respectively. When the ATP software response the fixed-time interrupts, it shall get the number of pulses and determine its correctness. Furthermore, the ATP software on two CPU shall synchronize their length of cycle, and ensure the difference is not exceeding the time of one interrupt. From the safety oriented, the ATP cycle time length only can be overestimated but cannot be underestimated.

#### Immediate Task Count

[iTC\_CC\_ATP-SwRS-0045]

VLEimpulseNb[**ATP\_INTERRUPT\_NB**]，存储每次触发中断时获取的VLE脉冲数。ATP软件在每次响应安全时钟的硬件中断后，需通过[VLE\_GetSafeTime](#VLE_GetSafeTime)接口获取VLE的脉冲数，将其存储在VLEimpulseNb数组中。

[VLEimpulseNb](#VLEimpulseNb)[**ATP\_INTERRUPT\_NB**] array stores the safe clock impulse number for every interrupt triggered. ATP shall obtain the impulse number through [VLE\_GetSafeTime](#VLE_GetSafeTime), and stores into the array.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0012]

[End]

[iTC\_CC\_ATP-SwRS-0756]

ATP上电后，在每次响应中断后将ImmediateCounter加1，作为中断计数器。

ImmediateCounter, as the interrupt counter, ATP shall accumulate 1 after each interrupt triggered.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0012]

[End]

[iTC\_CC\_ATP-SwRS-0046]

定时中断被激活后，在每次中断中对ImmediateNb，CycleSynchronized，Trace，Dt和CycleBiasNb进行更新：

* 如果当前是上电后第一个中断：ATP需设置ImmediateNb为0，并根据所在的CPU初始化Trace，VCP的时间标签Dt，以及中间变量m：

if ([DataPlugContent](#DataPlugContent).VLECpuId == **CPU1**)

[Dt](#Dt) = **CPU1\_DT\_INIT**

[Trace](#Trace) = **CPU1\_TRACE\_0** ^ [Dt](#Dt)

m = InversePDoperation(**CPU1\_TRACE\_0**, **CPU1\_TRACE\_N**)

else:

[Dt](#Dt) = **CPU2\_DT\_INIT**

[Trace](#Trace) = **CPU2\_TRACE\_0** ^ [Dt](#Dt)

m = InversePDoperation(**CPU2\_TRACE\_0**, **CPU2\_TRACE\_N**)

* 否则，如果ImmediateNb = 0，而[VitalTime](#VitalTime)与上个中断相比仍然未发生变化，则
* 将CycleBiasNb加1。
* 如果CycleBiasNb > 1，设置[CycleSynchronized](#CycleSynchronized)为**False**；
* 否则，如果ImmediateNb = 0，而[VitalTime](#VitalTime)与上个中断相比发生了变化，则
* 令ImmediateNb = 1；
* 令Trace = PDoperationDt([Trace](#Trace)，**Bi**[[ImmediateNb](#ImmediateNb)], [Dt](#Dt))
* 设置CycleBiasNb = 0，而且[CycleSynchronized](#CycleSynchronized)为**True**.
* 使用LockedImmediateCounter锁存此时的[ImmediateCounter](#ImmediateCounter)值，作为新周期初始的中断号。
* 否则，
* 将ImmediateNb的值加1；
* 令Trace = PDoperationDt([Trace](#Trace), **Bi**[[ImmediateNb](#ImmediateNb)], [Dt](#Dt))；
* 如果ImmediateNb > (**ATP\_INTERRUPT\_NB**-1)，则设置ImmediateNb = 0；并令Trace = PDoperation([Trace](#Trace)，m)，令Dt = PDoperation([Dt](#Dt), 0)

When the fixed-time interrupt triggered, ATP shall update the [ImmediateNb](#ImmediateNb)，[[CycleSynchronized](#CycleSynchronized)](#CycleSynchroized), [Trace](#Trace), [Dt](#Dt) and [CycleBiasNb](#CycleBiasNb).

* If it is the first interrupt after powered up, ATP shall set [ImmediateNb](#ImmediateNb) as zero，and initialize the [Trace](#Trace), [Dt](#Dt) (the dynamic time of VCP), and the middle variables m based on CPU.

if ([DataPlugContent](#DataPlugContent).VLECpuId == **CPU1**)

[Dt](#Dt) = **CPU1\_DT\_INIT**

[Trace](#Trace) = **CPU1\_TRACE\_0** ^ [Dt](#Dt)

m = InversePDoperation(**CPU1\_TRACE\_0**, **CPU1\_TRACE\_N**)

else:

[Dt](#Dt) = **CPU2\_DT\_INIT**

[Trace](#Trace) = **CPU2\_TRACE\_0** ^ [Dt](#Dt)

m = InversePDoperation(**CPU2\_TRACE\_0**, **CPU2\_TRACE\_N**)

* Or else:, If the [ImmediateNb](#ImmediateNb) is zero, but the [VitalTime](#VitalTime) has not changed comparing to the previous interrupt, then:

[CycleBiasNb](#CycleBiasNb) = [CycleBiasNb](#CycleBiasNb) + 1

if ([CycleBiasNb](#CycleBiasNb) > 1)

[[CycleSynchronized](#CycleSynchronized)](#CycleSynchroized) = **False**

* Or else, If [ImmediateNb](#ImmediateNb) is zero, and the [VitalTime](#VitalTime) has changed comparing to the previous interrupt, then:

[ImmediateNb](#ImmediateNb) = 1

[Trace](#Trace) = PDoperationDt([Trace](#Trace)，**Bi**[[ImmediateNb](#ImmediateNb)], [Dt](#Dt))

[CycleBiasNb](#CycleBiasNb) = 0

[[CycleSynchronized](#CycleSynchronized)](#CycleSynchroized) = **True**

[LockedImmediateCounter](#LockedImmediateCounter) **=** [ImmediateCounter](#ImmediateCounter)

* Otherwise, set:

[ImmediateNb](#ImmediateNb) = [ImmediateNb](#ImmediateNb) + 1

[Trace](#Trace) = PDoperationDt([Trace](#Trace), **Bi**[[ImmediateNb](#ImmediateNb)], [Dt](#Dt))

* and if the [ImmediateNb](#ImmediateNb) is greater than (**ATP\_INTERRUPT\_NB**-1), then:

[ImmediateNb](#ImmediateNb) = 0

[Trace](#Trace) = PDoperation([Trace](#Trace)，m)

[Dt](#Dt) = PDoperation([Dt](#Dt), 0)

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0012], [iTC\_CC\_ATP\_SwHA-0002], [iTC\_CC\_ATP\_SwHA-0004]

[End]

**NOTES：**

其中**CPU1\_TRACE\_0**，**CPU2\_TRACE\_0**，**CPU1\_TRACE\_N**，**CPU2\_TRACE\_N**，**CPU1\_DT\_INIT**，**CPU2\_DT\_INIT**，以及数组**Bi**[**ATP\_INTERRUPT\_NB**]为VCP事先分配好的特征值。而PDoperation()表示不带Dt的PD运算；PdoperationDt()为带Dt的PD运算；InversePDoperation()为PD运算的逆运算。

The numbers of data is classified by the VCP tool beforehand which includes: **CPU1\_TRACE\_0**, **CPU2\_TRACE\_0**, **CPU1\_TRACE\_N**, **CPU2\_TRACE\_N**, **CPU1\_DT\_INIT**, **CPU2\_DT\_INIT**, and **Bi**[**ATP\_INTERRUPT\_NB**]. and PDoperation() represents the PD calculation without [Dt](#Dt)；PDoperationDt() stands for the PD calculation with [Dt](#Dt) ; InversePDoperation() is regarded as PD inverse calculation.

#### Deferred Task Control

[iTC\_CC\_ATP-SwRS-0535]

SafeTimerFailed，判断硬件定时中断是否正确。在主任务中检查每个相邻中断中锁存的VLE安全时钟脉冲数是否在误差范围[**MIN\_TIMER\_IMPULSE\_NB**, **MAX\_TIMER\_IMPULSE\_NB**]内。

* 一旦判断SafeTimerFailed为**True**，则始终保持为**True**（只有重启ATP才能缓解）；
* 否则，若测得脉冲数在上述范围内，则设置SafeTimerFailed为**False**;
* 否则，设置SafeTimerFailed为**True**。

The [SafeTimerFailed](#SafeTimerFailed) defines whether the fix-time interrupt for hardware is correct or not. ATP shall check whether the safe clock impulse number with the adjacent interrupt is within the error range [**MIN\_TIMER\_IMPULSE\_NB**, **MAX\_TIMER\_IMPULSE\_NB**].

* Once the [SafeTimerFailed](#SafeTimerFailed) was **True**, ATP shall keep it as **True** unless the system is rebooted.
* Or else:, if the impulse number is within the above-mentioned range, ATP shall set [SafeTimerFailed](#SafeTimerFailed) as **False**
* Otherwise, it will set [SafeTimerFailed](#SafeTimerFailed) as **True**.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0012], [iTC\_CC\_ATP\_SwHA-0172]

[End]

[iTC\_CC\_ATP-SwRS-0047]

VitalTime，ATP主任务维护的当前周期序号。上电后从0开始，每周期递增加1。当主任务在执行完成本周期的所有工作后，监控中断任务是否执行完成，即[ImmediateCounter](#ImmediateCounter)和[LockedImmediateCounter](#LockedImmediateCounter)的差值是否大于等于（**ATP\_INTERRUPT\_NB** -1）：

* 若是，则表明主周期执行完成：
* 将VitalTime送给另一个CPU模块；
* 将[Trace](#Trace)(k)和[Dt](#Dt)(k)作为校核字送给VIOM进行校验。
* 在本周期最后，设置

[VitalTime](#VitalTime) = [VitalTime](#VitalTime)(k-1) + 1

* 否则，继续等待。

The [VitalTime](#VitalTime) stand for the current cycle of ATP deferred task. After power up, it starts from zero and increase one each cycle. When all the work is executed in the main task, ATP detects whether the interrupt task is over, i.e. the difference between [ImmediateCounter](#ImmediateCounter) and [LockedImmediateCounter](#LockedImmediateCounter) is equal to or larger than (**ATP\_INTERRUPT\_NB** -1).

* If it is so, it shows that the main task in this cycle finishes. Then ATP shall:
* send the [VitalTime](#VitalTime) to the other CPU,
* and send [Trace](#Trace) and [Dt](#Dt) to VIOM to check,
* and at the end of this cycle, set

[VitalTime](#VitalTime) = [VitalTime](#VitalTime)(k-1) + 1

* Otherwise, keep waiting.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0115], [iTC\_CC\_ATP\_SwHA-0004]

[End]

[iTC\_CC\_ATP-SwRS-0048]

当满足下列条件时，执行新的主周期：

* [[CycleSynchronized](#CycleSynchronized)](#CycleSynchroized) = **True**,
* and [TOC\_VitalTime](#TOC_VitalTime) == [VitalTime](#VitalTime)(k)，表示另一个CPU执行已执行完成上个周期的任务。

如果不满足上述条件，则不允许执行新周期，CPU1的ATP在VLE板的LED上显示**ERR\_SYNCH**信息。

ATP executes the new cycle DeferredTask when below conditions fulfilled:

* [[CycleSynchronized](#CycleSynchronized)](#CycleSynchroized) is **True**,
* and the [TOC\_VitalTime](#TOC_VitalTime) get from the other CPU is equal to [VitalTime](#VitalTime), representing that the other CPU finished to execute the task in the previous cycle.

If above condition does not fulfill, ATP shall prohibit to execute, and CPU1 shows the message **ERR\_SYNCH** in the LED of VLE board.

#Category= Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0831], [iTC\_CC\_ATP\_SwHA-0003], [iTC\_CC-SyAD-1423]

[End]

说明: 说明: F5-3_synchronize_2_cpu.emf

Figure 5‑23 Synchronize cycle sequence between two CPUs

**NOTES**:

如Figure 5‑23为双CPU模块周期同步的示意图。如果在Cycle N时，CPU1的时钟快于CPU2，则CPU1的中断计数器提前数到**ATP\_INTERRUPT\_NB**，则向CPU2发送CPU1\_VitalTime(N)信息，但此时CPU2的中断计数器尚未到新周期，则此时两个CPU模块均不能进入Cycle(N+1)。只有等CPU2接到下个周期的开始标志之后，才能进入Cycle(N+1)，同时，发送CPU2\_VitalTime(N)信号给CPU1，通知CPU1进入新周期。而此时，CPU1则忽略一个中断时长。即相当于CPU1“等待”了CPU2一段时间，起到了同步作用。

As shown in Figure 5‑23, this is the demonstration of the process for cycle synchronization between two CPU modules. In cycle N, if the clock of CPU1 was faster than CPU2, then the interrupt counter of CPU1 counts to **ATP\_INTERRUPT\_NB** and sends the CPU1\_VitalTime(N) information to CPU2. However, at this moment the interrupt counter of CPU2 still do not achieve the new cycle, so both of the CPU module cannot enter into cycle(N+1). Only when the CPU2 has get all interrupt finished signal, then sends the VitalTime(N) to CPU1, and the both CPUs are enter the cycle(N+1) together. During this process, CPU1 has ignored one interrupt period to wait the CPU2.

这种趋向于“等待”的同步方式，会避免时钟“变快”的危险。就是说，可能将(**ATP\_CYCLE\_TIME**+**ATP\_INTERRUPT\_TIME**)时间内测得的位移，除以**ATP\_CYCLE\_TIME**时间，则得到的速度是比实际要大的，即过估了列车速度，从而保证安全。

This kind of "waiting" synchronous way will avoid the danger that the clock is getting faster and faster. That is to say, we may use the measured movement in the period (**ATP\_CYCLE\_TIME** + **ATP\_INTERRUPT\_TIME**) divide a fixed **ATP\_CYCLE\_TIME**, then we can get speed which is higher than the actual value, meaning that we over-evaluate the train speed so as to ensure the safety.

CPU1和CPU2的周期误差不能超过1次中断时长**ATP\_INTERRUPT\_TIME**，如果超过了该时间，则设置[[CycleSynchronized](#CycleSynchronized)](#CycleSynchroized)为**False**，而ATP将不会进入执行新的主周期。从而导致VIOM切断对车辆的输出。

The cycle error between CPU1 and CPU2 cannot exceed one **ATP\_INTERRUPT\_TIME**. If it exceeded this period, the [[CycleSynchronized](#CycleSynchronized)](#CycleSynchroized) shall be set as **False**, and the ATP will not execute continuously so that the VIOM will cut off the output to the train.

[iTC\_CC\_ATP-SwRS-0589]

为确保ATP与CCNV的周期同步，ATP应当每**CCNV\_CYCLE\_TIME**触发一次DVCOM-2板的中断。

#Category= Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-1134]

[End]

#### RTC time

[iTC\_CC\_ATP-SwRS-0446]

RTCtime，ATP维护的非安全时钟：

* ATP软件在初始化时从VLE-2后板上获取RTC时钟信息；
* 之后，ATP软件每秒钟将该RTC时钟加1；
* 但如果RTC时钟与来自CCNV的NTP时间差超过**MAX\_NTP\_TIME\_ERROR**，则使用NTP时间更新RTC时间。

ATP software shall maintain the RTC time for non-vital functions.

* In initialization, ATP software get RTC time from VLE-2 board;
* And then, ATP software updates the RTC time every second;
* And if the difference between RTC time ATP used and the NTP time CCNV sent is greater than **MAX\_NTP\_TIME\_ERROR**, ATP shall reset the RTC time as NTP time.

if (Initialization)

[RTCtime](#RTCtime) = [VLE\_RTCtime](#VLE_RTCtime)

elif (([ATOcontrolTimeValid](#ATOcontrolTimeValid)(k) == **True**)

and ([NonVitalRequest](#NonVitalRequest).NtpTime != **None**)

and (|[NonVitalRequest](#NonVitalRequest).NtpTime - [RTCtime](#RTCtime)(k-1)| > **MAX\_NTP\_TIME\_ERROR**))

[RTCtime](#RTCtime) = [NonVitalRequest](#NonVitalRequest).NtpTime

else:

[RTCtime](#RTCtime) = Time.Update()

其中Time.Update()意为ATP软件每秒钟将RTC时间加1。

The Time.Update() means ATP software shall update the RTC time every second.

#Category=Functional

#Contribution=SIL0

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0760], [iTC\_CC-SyAD-0761], [iTC\_CC-SyAD-1007]

[End]

### Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Safety Critical | | | Observable | | Logical Type |
| [Dt](#Dt) | | √ | √ | | NUMERIC\_32 | |
| [ImmediateNb](#ImmediateNb) | | √ | × | | NUMERIC\_32 | |
| [RTCtime](#RTCtime) | | × | √ | | NUMERIC\_32 | |
| [SafeTimerFailed](#SafeTimerFailed) | | √ | √ | | BOOLEAN | |
| [Trace](#Trace) | | √ | √ | | NUMERIC\_32 | |
| [VitalTime](#VitalTime) | | √ | √ | | NUMERIC\_32 | |

## F83-Manage Synchronization between two CPUs

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| [CBIvariantRequest](#CBIvariantRequest) | External | Interface with CI Radio |
| [CCsynchroReport](#CCsynchroReport) | External | Interface with Distant ATP |
| [CCvariantReport](#CCvariantReport) | External | Interface with CI Radio |
| [CIsetting](#CIsetting) | External | Interface with PSD |
| [LocReport](#LocReport) | External | Interface with ZC |
| [TOC\_CBIvariantRequest](#TOC_CBIvariantRequest) | External | Interface between two CPUs |
| [TOC\_CCsyncReport](#TOC_CCsyncReport) | External | Interface between two CPUs |
| [TOC\_CCvariantReport](#TOC_CCvariantReport) | External | Interface between two CPUs |
| [TOC\_CIsetting](#TOC_CIsetting) | External | Interface between two CPUs |
| [TOC\_LocReport](#TOC_LocReport) | External | Interface between two CPUs |
| [TOC\_VersionReport](#TOC_VersionReport) | External | Interface between two CPUs |
| [TOC\_VIOM2VitalOut](#TOC_VIOM2VitalOut) | External | Interface between two CPUs |
| [VersionFromCCreport](#VersionFromCCreport) | External | Interface with LC |
| [VIOM1VitalOut](#VIOM1VitalOut) | External | F71-Outputs to Rolling-stock |
| [VIOM2VitalOut](#VIOM2VitalOut) | External | F71-Outputs to Rolling-stock |

### Processing

如Figure 5‑24所示，同一块VLE-2板上的两套ATP软件，在计算出给VIOM，冗余ATP，ZC和LC的输出结果后，需通过双口RAM交换该结果，并将对方的计算结果和自己的结果合并后发送给相关外部系统。

Refer to Figure 5‑24, the two set of ATP software in the same VLE-2 board will calculate the output results to VIOM, redundant ATP, [ZC](#ZC) or LC, and those data will exchanged by DPRAM. Then the output will merge into one pack of data and send to CCNV.

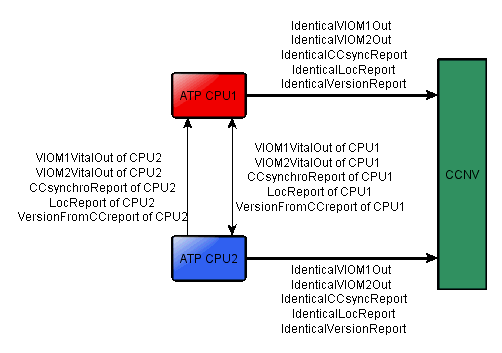


Figure 5‑24 Manage synchronization between two CPUs

[iTC\_CC\_ATP-SwRS-0050]

车载ATP软件在每个周期运行时，需要将VLE-2板两个CPU模块生成的安全输出命令合并组成一帧后发送给CCNV，由其转发给VIOM。在每周期生成运算结果后，ATP软件通过双口RAM将运算结果[VIOM1VitalOut](#VIOM1VitalOut)和[VIOM2VitalOut](#VIOM2VitalOut)发送给另一个CPU，并获取来自另一个CPU的运算结果[TOC\_VIOM1VitalOut](#TOC_VIOM1VitalOut)和[TOC\_VIOM2VitalOut](#TOC_VIOM2VitalOut)。

In every cycle, ATP combines the vital outputs generated by itself and from the other CPU into one frame and sends it to CCNV, who will transmit the frame to VIOM. ATP will send [VIOM1VitalOut](#VIOM1VitalOut) and [VIOM2VitalOut](#VIOM2VitalOut) to the other CPU through dual-ports RAM, and receive the [TOC\_VIOM1VitalOut](#TOC_VIOM1VitalOut) and [TOC\_VIOM2VitalOut](#TOC_VIOM2VitalOut) from the other CPU.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0007], [iTC\_CC\_ATP\_SwHA-0005]

[End]

[iTC\_CC\_ATP-SwRS-0051]

车载ATP软件将[VIOM1VitalOut](#VIOM1VitalOut)和[TOC\_VIOM1VitalOut](#TOC_VIOM1VitalOut)按照ST\_VIOM\_OUT结构组合成IdenticalVIOM1Out，将[VIOM2VitalOut](#VIOM2VitalOut)和[TOC\_VIOM2VitalOut](#TOC_VIOM2VitalOut)按照ST\_VIOM\_OUT结构组合成IdenticalVIOM2Out。

Based on ST\_VIOM\_OUT structure, ATP combines [VIOM1VitalOut](#VIOM1VitalOut) and [TOC\_VIOM1VitalOut](#TOC_VIOM1VitalOut) as [IdenticalVIOM1Out](#IdenticalVIOM1Out), while based on the same structure ST\_VIOM\_OUT, ATP combines [VIOM2VitalOut](#VIOM2VitalOut) and [TOC\_VIOM2VitalOut](#TOC_VIOM2VitalOut) as [IdenticalVIOM2Out](#IdenticalVIOM2Out).

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC-SyAD-0007], [iTC\_CC\_ATP\_SwHA-0005]

[End]

[iTC\_CC\_ATP-SwRS-0579]

IdenticalLocReport，ATP两CPU同步后的发送给ZC的位置报告信息。当两CPU各自计算完成[LocReport](#LocReport)后，需将其通过双口RAM发送给对方；并接收来自对方的[TOC\_LocReport](#TOC_LocReport)，共同组成IdenticalLocReport，再发送给CCNV，由其转发给ZC。IdenticalLocReport的生成规则如下：

* ST\_LOC\_REPORT中除两重SACEM校核字外，其余均采用本CPU的计算结果；
* 对于VitalChecksum\_1，采用CPU1的数据进行计算；
* 对于VitalChecksum\_2，采用CPU2的数据进行计算。

[IdenticalLocReport](#IdenticalLocReport), the location report after merging two CPU’s results. When the two CPUs complete [LocReport](#LocReport) calculation, need to send each other through the dual-port RAM; and receive the [TOC\_LocReport](#TOC_LocReport) from the other. ATP shall combine the two reports as an [IdenticalLocReport](#IdenticalLocReport), according to the following rules:

* The variables without vital checksums in ST\_LOC\_REPORT, shall use the values calculated by itself.
* For the VitalChecksum\_1, ATP shall calculate using the values from CPU1.
* For the VitalChecksum\_2, ATP shall calculate using the values from CPU2.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0005]

[End]

**NOTES**:

正常情况下，来自CPU1和CPU2的计算结果应当相同。但是，由于CPU1和CPU2分别通过VPB的上下模块获取里程计数据，由于获取时机的不同步（VLE板两个CPU的中断是各自独立的），可能会导致ATP软件在计算列车位移、速度及其定位上的细微差。因此，需ATP上下模块将获取到的VPB输入信息进行同步（SwRS-0590），之后再计算。若计算出的两个模块的输出结果仍然不一致，则接收方将判断出两路校核字计算结果不一致，从而拒绝该消息。

Normally, the calculated results between CPU1 and CPU2 should be the same – if not, the receiver will get the wrong checksum calculation and thus reject the message. However, due to the independent interrupts of two CPUs, the moment ATP software reading the odometer information from VPB’s registers is not simultaneous, which lead to unavoidable biases in movement or speed calculation by the two ATP software. Therefore, ATP need to synchronize the VPB inputs (SwRS-0590) and then calculate with [IdenticalLockedOdometer](#IdenticalLockedOdometer).

[iTC\_CC\_ATP-SwRS-0580]

IdenticalVersionReport，ATP两CPU同步后的发送给LC的版本报告信息。

当两CPU各自计算完成[VersionFromCCreport](#VersionFromCCreport)后，需将其通过双口RAM发送给对方；并接收来自对方的[TOC\_VersionReport](#TOC_VersionReport)，组合生成IdenticalVersionReport，再发送给CCNV，由其转发给LC。IdenticalVersionReport的生成规则如下：

* ST\_VERSION\_REPORT中除安全校核字外，其余变量均采用本CPU的计算结果；
* 对于VitalChecksum\_1，采用CPU1的数据进行计算；
* 对于VitalChecksum\_2，采用CPU2的数据进行计算。

[IdenticalVersionReport](#IdenticalVersionReport), the version report after merging two CPU’s results. When the two CPUs complete [VersionFromCCreport](#VersionFromCCreport) calculation, need to send each other through the dual-port RAM; and receive the [TOC\_VersionReport](#TOC_VersionReport) from the other. ATP shall combine the two reports as an [IdenticalVersionReport](#IdenticalVersionReport), according to the following rules:

* The variables without vital checksums in ST\_VERSION\_REPORT, shall use the values calculated by itself.
* For the VitalChecksum\_1, ATP shall calculate using the values from CPU1.
* For the VitalChecksum\_2, ATP shall calculate using the values from CPU2.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0005]

[End]

[iTC\_CC\_ATP-SwRS-0581]

IdenticalCCsyncReport，ATP两CPU同步后的发送给冗余ATP的同步信息。

当两CPU各自计算完成[CCsynchroReport](#CCsynchroReport)后，需将其通过双口RAM发送给对方；并接收来自对方的[TOC\_CCsyncReport](#TOC_CCsyncReport)组合生成IdenticalCCsyncReport，再发送给CCNV，由其转发给冗余ATP。IdenticalCCsyncReport的生成规则如下：

* ST\_SYNCHRO\_REPORT中除安全校核字外，其余变量均采用本CPU的计算结果；
* 对于VitalChecksum\_1，采用CPU1的数据进行计算；
* 对于VitalChecksum\_2，采用CPU2的数据进行计算。

[IdenticalCCsyncReport](#IdenticalCCsyncReport), the redundant ATP report after merging two CPU’s results. When the two CPUs complete [CCsynchroReport](#CCsynchroReport) calculation, need to send each other through the dual-port RAM; and receive the [TOC\_CCsyncReport](#TOC_CCsyncReport) from the other. ATP shall combine the two reports as an [IdenticalCCsyncReport](#IdenticalCCsyncReport), according to the following rules:

* The variables without vital checksums in ST\_SYNCHRO\_REPORT, shall use the values calculated by itself.
* For the VitalChecksum\_1, ATP shall calculate using the values from CPU1.
* For the VitalChecksum\_2, ATP shall calculate using the values from CPU2.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0005]

[End]

[iTC\_CC\_ATP-SwRS-0588]

IdenticalCIsetting[**MAX\_CONNECTED\_PSD\_NB**],ATP两CPU同步后的发送给联锁PSD控制命令。

当两CPU各自计算完成[CIsetting](#CIsetting)后，需将其通过双口RAM发送给对方；并接收来自对方的[TOC\_CIsetting](#TOC_CIsetting)组合生成IdenticalCIsetting，再发送给CCNV，由其转发给冗余ATP。IdenticalCIsetting的生成规则如下：

* ST\_CI\_SETTING除校核字外的变量，采用本CPU的计算结果；
* 对于FSFB2通信协议中的校核字1，采用CPU1的数据进行计算；
* 对于FSFB2通信协议中的校核字2，采用CPU2的数据进行计算。

[IdenticalCIsetting](#IdenticalCIsetting), the PSD control message after merging two CPU’s results. When the two CPUs complete [CIsetting](#CIsetting) calculation, need to send each other through the dual-port RAM; and receive the [TOC\_CIsetting](#TOC_CIsetting) from the other. ATP shall combine the two reports as an [IdenticalCIsetting](#IdenticalCIsetting), according to the following rules:

* The variables in ST\_CI\_SETTING, shall use the values calculated by itself.
* For the checkword 1 in FSFB2 protocol, ATP shall calculate using the values from CPU1.
* For the checkword 2 in FSFB2 protocol, ATP shall calculate using the values from CPU2.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0005]

[End]

[iTC\_CC\_ATP-SwRS-0746]

IdenticalCBIvariantRequest，ATP两CPU同步后的发送给联锁的变量请求信息。

当两CPU各自计算完成[CBIvariantRequest](#CBIvariantRequest)后，需将其通过双口RAM发送给对方；并接收来自对方的[TOC\_CBIvariantRequest](#TOC_CBIvariantRequest)组合生成IdenticalCBIvariantRequest，再发送给CCNV，由其转发给联锁。IdenticalCBIvariantRequest的生成规则如下：

* 除安全校核字外，均采用本CPU的计算结果；
* 对于VitalChecksum\_1，采用CPU1的数据进行计算；
* 对于VitalChecksum\_2，采用CPU2的数据进行计算。

[IdenticalCBIvariantRequest](#IdenticalCBIvariantRequest), the CBI variants request after merging two CPU’s results. When the two CPUs complete [CBIvariantRequest](#CBIvariantRequest)calculation, need to send each other through the dual-port RAM; and receive the [TOC\_CCvariantReport](#TOC_CCvariantReport) from the other. ATP shall combine the two reports as an [IdenticalCBIvariantRequest](#IdenticalCBIvariantRequest), according to the following rules:

* The variables without vital checksums, shall use the values calculated by itself.
* For the VitalChecksum\_1, ATP shall calculate using the values from CPU1.
* For the VitalChecksum\_2, ATP shall calculate using the values from CPU2.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0005]

[End]

[iTC\_CC\_ATP-SwRS-0747]

IdenticalCCvariantReport，ATP两CPU同步后的发送给联锁的Overlap解锁信息。

当两CPU各自计算完成[CCvariantReport](#CCvariantReport)后，需将其通过双口RAM发送给对方；并接收来自对方的[TOC\_CCvariantReport](#TOC_CCvariantReport)组合生成IdenticalCCvariantReport，再发送给CCNV，由其转发给联锁。IdenticalCCvariantReport的生成规则如下：

* 除安全校核字外，均采用本CPU的计算结果；
* 对于VitalChecksum\_1，采用CPU1的数据进行计算；
* 对于VitalChecksum\_2，采用CPU2的数据进行计算。

[IdenticalCCvariantReport](#IdenticalCCvariantReport), the CC overlap releasable report after merging two CPU’s results. When the two CPUs complete [CCvariantReport](#CCvariantReport)calculation, need to send each other through the dual-port RAM; and receive the [TOC\_CCvariantReport](#TOC_CCvariantReport) from the other. ATP shall combine the two reports as an [IdenticalCCvariantReport](#IdenticalCCvariantReport), according to the following rules:

* The variables without vital checksums, shall use the values calculated by itself.
* For the VitalChecksum\_1, ATP shall calculate using the values from CPU1.
* For the VitalChecksum\_2, ATP shall calculate using the values from CPU2.

#Category=Functional

#Contribution=SIL4

#Allocation=ATP Software

#Source=[iTC\_CC\_ATP\_SwHA-0005]

[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| [IdenticalCBIvariantRequest](#IdenticalCBIvariantRequest) | √ | × | ST\_CBI\_VARIANT\_REQUEST |
| [IdenticalCCsyncReport](#IdenticalCCsyncReport) | √ | × | ST\_SYNCHRO\_REPORT |
| [IdenticalCCvariantReport](#IdenticalCCvariantReport) | √ | × | ST\_CC\_VARIANT\_REPORT |
| [IdenticalCIsetting](#IdenticalCIsetting) | √ | × | ST\_CI\_SETTING |
| [IdenticalLocReport](#IdenticalLocReport) | √ | × | ST\_LOC\_REPORT |
| [IdenticalVIOM1Out](#IdenticalVIOM1Out) | √ | × | ST\_VIOM\_OUT |
| [IdenticalVIOM2Out](#IdenticalVIOM2Out) | √ | × | ST\_VIOM\_OUT |
| [IdenticalVersionReport](#IdenticalVersionReport) | √ | × | ST\_VERSION\_REPORTST\_VERSION\_REPORT |

# Appendices

## Project Configuration of ATPsetting

The project configurable vital settings are shown in Table 6‑1.

Table 6‑1 Project setting parameters structe

|  |  |  |
| --- | --- | --- |
| Identification | Type | Unit |
| BeaconPairMaxDistance | NUMERIC\_32 | mm |
| BlockModeThroughRadio | BOOLEAN | - |
| BMinitAreaLength | NUMERIC\_32 | mm |
| BrakingMinAcc | NUMERIC\_32 | mm/s^2 |
| BrakingStartAcc | NUMERIC\_32 | mm/s^2 |
| CCcoreEnd2BeaconAntennaDistance[**0**..**2**] | NUMERIC\_32 | mm |
| CCcoreOdoCogIncreasing[**0**..**2**] | NUMERIC\_32 | -1, 0, 1 |
| DoorTrainLockingSpeed | NUMERIC\_32 | mm/s |
| DoorTrainOpeningLimitSpeed | NUMERIC\_32 | mm/s |
| DoorTrainUnlockingSpeed | NUMERIC\_32 | mm/s |
| EBguaranteedAccNormalGrip | NUMERIC\_32 | mm/s^2 |
| EBguaranteedAccReducedGrip | NUMERIC\_32 | mm/s^2 |
| EBtractionCutoffLatency | NUMERIC\_32 | cycle |
| EBtractionToBrakingLatency | NUMERIC\_32 | cycle |
| End2Orientation | ENUM\_DOT | - |
| EOAmaxDistance | NUMERIC\_32 | mm |
| EOAvalidityTime | NUMERIC\_32 | cycle |
| EvacuationStationAreaLength | NUMERIC\_32 | mm |
| EvacuationTrainEnd | BOOLEAN | - |
| LCloophourValidityTime | NUMERIC\_32 | cycle |
| LocationBeaconValidityDistance | NUMERIC\_32 | mm |
| LocationMaxUncertaintyConfirmed | NUMERIC\_32 | mm |
| LocationMaxUncertaintyNotConfirmed | NUMERIC\_32 | mm |
| LocationTrainLength | NUMERIC\_32 | mm |
| LocReportEmissionPeriod | NUMERIC\_32 | cycle |
| LocReportValidityTime | NUMERIC\_32 | cycle |
| MaxGradientAcc | NUMERIC\_32 | mm/s^2 |
| MaxMotionPerCycle | NUMERIC\_32 | mm |
| MemLocAuth | BOOLEAN |  |
| MemLocLimitSpeed | NUMERIC\_32 |  |
| MemLocValidityDistance | NUMERIC\_32 |  |
| MeterCaliMaxMinCalibration[0…1][**0**…**CALI\_TABLE\_LENGTH**] | NUMERIC\_32 | mikron |
| MTdistance | NUMERIC\_32 | mm |
| MTimmoBehaviourAtFS | ENUM\_SAFE\_IMMO\_BEHAVIOUR | - |
| MTtimeout | NUMERIC\_32 | cycle |
| MPauthAltitudeMaxErrorEnergy | NUMERIC\_32 | mm^2/s^2 |
| MPauthImmoBehaviourAtFS | ENUM\_SAFE\_IMMO\_BEHAVIOUR |  |
| MPauthLimitSpeed | NUMERIC\_32 | mm/s |
| MPinhibitionLimitSpeed[**MAX\_RM\_CONDITION\_NB**] | NUMERIC\_32 | mm/s |
| MPnotAuthDistWithoutMotionAvailable | NUMERIC\_32 | mm |
| MPnotAuthLimitDistance | NUMERIC\_32 | mm |
| MPnotAuthLimitSpeed | NUMERIC\_32 | mm/s |
| NUDEdistanceWithoutMotionAvailable | NUMERIC\_32 | mm |
| NUDEimmoBehaviourAtFS | ENUM\_SAFE\_IMMO\_BEHAVIOUR | - |
| NUDElimitDistance | NUMERIC\_32 | mm |
| NUDEtrainStopDurationBeforeEBrelease | NUMERIC\_32 | cycle |
| OdoCaliCogCounterMax | NUMERIC\_32 |  |
| OdoCaliCogCounterMin | NUMERIC\_32 |  |
| OdoCaliDefaultCogLengthMax | NUMERIC\_32 | mikron |
| OdoCaliDefaultCogLengthMin | NUMERIC\_32 | mikron |
| OdoInitTimeout | NUMERIC\_32 | cycle |
| OdoLockedAxleDisablingLatency | NUMERIC\_32 | cycle |
| OdoLockedAxleEnablingLatency | NUMERIC\_32 | cycle |
| OdoLockedAxleThresholdSpeed | NUMERIC\_32 | mm/s |
| OdoLockedAxleTimeout | NUMERIC\_32 | cycle |
| OdoMaxCogOnCycle | NUMERIC\_32 | - |
| OdoMaxCogOnInterrupt | NUMERIC\_32 | - |
| OdoMaxWheelMotion | NUMERIC\_32 | mm |
| OdoMinDistAfterSenseChange | NUMERIC\_32 | mm |
| OdoNotOnMotorizedAxle | BOOLEAN | - |
| OdoTestContradictionDuration | NUMERIC\_32 | cycle |
| PolarizedTrain | BOOLEAN | - |
| PSDcommCycle | NUMERIC\_32 | cycle |
| PSDopeningCommandValidityTime | NUMERIC\_32 | cycle |
| PSDstatusValidityTime | NUMERIC\_32 | cycle |
| RadarApplied | BOOLEAN | - |
| RadarDeviationAboveThreshold | NUMERIC\_32 | 0.1% |
| RadarDeviationBelowThreshold | NUMERIC\_32 | mm/s |
| RadarSpeedThreshold | NUMERIC\_32 | mm/s |
| ReverseDistWithoutMotionAvailable | NUMERIC\_32 | mm |
| ReverseLimitCount | NUMERIC\_32 | - |
| ReverseLimit[**MAX\_REVERSE\_LIMIT\_STEP**] | | |
| .Distance | NUMERIC\_32 | mm |
| .Speed | NUMERIC\_32 | mm/s |
| SafetyParameterVersion | NUMERIC\_32 | - |
| SlidingCoefficient | NUMERIC\_32 | - |
| SlidingExcessTime | NUMERIC\_32 | cycle |
| SlidingGripRecoveryTime | NUMERIC\_32 | cycle |
| SlidingStartAcc | NUMERIC\_32 | mm/s^2 |
| SlidingStopAcc | NUMERIC\_32 | mm/s^2 |
| SlidingTimeout | NUMERIC\_32 | cycle |
| SlippingCoefficient | NUMERIC\_32 | - |
| SlippingExcessTime | NUMERIC\_32 | cycle |
| SlippingGripRecoveryTime | NUMERIC\_32 | cycle |
| SlippingStartAcc | NUMERIC\_32 | mm/s^2 |
| SlippingStopAcc | NUMERIC\_32 | mm/s^2 |
| SlippingTimeout | NUMERIC\_32 | cycle |
| TractionMaxAccCount | NUMERIC\_32 | - |
| TractionMaxAcc[TractionMaxAccCount] | | |
| .Speed | NUMERIC\_32 | mm/s |
| .Value | NUMERIC\_32 | mm/s^2 |
| TractionStartAcc | NUMERIC\_32 | mm/s^2 |
| TrainTypeId | NUMERIC\_32 | - |
| TSRvalidityTime | NUMERIC\_32 | cycle |
| TSRdefaultLimitSpeed | NUMERIC\_32 | mm/s |
| VariantsBMALSpresenceTimer | NUMERIC\_32 | cycle |
| VariantsBMfullValidityTime | NUMERIC\_32 | cycle |
| VariantsBMlowValidityTime | NUMERIC\_32 | cycle |
| VariantsBMproductionLatencyBeacon | NUMERIC\_32 | cycle |
| VariantsBMproductionLatencyRadio | NUMERIC\_32 | cycle |
| VariantsBMradioPriorityDelay | NUMERIC\_32 | cycle |
| VariantsCBTCvalidityTime | NUMERIC\_32 | cycle |
| VersionsValidityTime | NUMERIC\_32 | cycle |

## Carborne Controller Constants

Table 6‑2描述了CC系统使用的默认常数，这些参数不随项目而改变。

Table 6‑2 shows the constants used by ATP software, which are commonly used for all iTC projects.

Table 6‑2 Carborne Controller Constants

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | Value | Unit | Description |
| **ABSCISSA\_TO\_HALF\_METER** | 500 | - | LocReport中的坐标单位转换常数 |
| **ATP\_CYCLE\_TIME** | 0.1 | second | Cycle length for ATP software |
| **ATP\_CYCLE\_TIME\_MS** | 100 | millisecond | Cycle length for ATP software |
| **ATP\_INTERRUPT\_NB** | 50 | - | Interrupt number of one cycle |
| **ATP\_INTERRUPT\_TIME** | 2 | millisecond | Period of the fixed interrupt |
| **ATP\_MESSAGE\_MAX\_DELAY** | 2^21 |  | Constant of modular subtraction |
| **AVERAGE\_ACCELERATION\_NB** | 32 | cycle | Cycles number for calculate average acceleration |
| **C\_TTIS** | 15 | minute | 烧录ATP软件的时间  Period for setup the ATP software |
| **CC1\_INIT\_TIME** | 0 | - | CC1的起始周期号  Initialization loop hour for CC in train **END\_1** |
| **CC1\_MAX\_TIME** | 2^29-1 | - | CC1的最大周期号  Maximum loop hour for CC in train **END\_1** |
| **CC2\_INIT\_TIME** | 2^29 | - | CC2的起始周期号  Initialization loop hour for CC in train **END\_2** |
| **CC2\_MAX\_TIME** | 2^30-1 |  | CC2的最大周期号  Maximum loop hour for CC in train **END\_2** |
| **CCNV\_CYCLE\_TIME** | 100 | millisecond | CCNV的周期时间  Cycle time of CCNV software |
| **CCNV\_VALIDITY\_CYCLES** | 6 | cycle | CCNV消息有效的周期数  Validity time of CCNV message |
| **COUPLE\_VIOM\_IN\_CAB1** | 2 | - | 车头1的VIOM个数 |
| **COUPLE\_VIOM\_IN\_CAB2** | 2 | - | 车头2的VIOM个数 |
| **END\_1** | 1 | - | Train front end 1 |
| **END\_2** | 2 | - | Train front end 2 |
| **FILTERED\_ACCELERATION\_NB** | 4 | cycle | Cycles number for calculate filtered acceleration |
| **FSFB2\_MESSAGE\_TIMEOUT** | 5 | - | Maximum communication cycle of FSFB2 message invalid |
| **HIGH\_LEVEL** | 1 | - | There is a signal from the sensor |
| **INIT\_AVAIL\_MAX\_TIME** | 45 | second | 从上电到可用的时间  Period length from power on to ATP software available |
| **INVALID\_LC\_DATE** | -(2^31)+1 | - | 无效的LC时间值  Invalid LC date |
| **INVALID\_LOOP\_HOUR** | -(2^20) | - | 无效的LoopHour值  Invalid loop hour |
| **KMPH\_TO\_MMPS** | 2500/9 | - | Convert unit from kph to mm/s |
| **LOW\_LEVEL** | 0 | - | No signal from the sensor |
| **MAX\_ATP\_LOOP\_HOUR** | 2^29 | - | 最大loop hour差值  Maximum difference of the loop hour |
| **MAX\_BEACON\_DATA\_SIZE** | 29 | - | 信标数据长度  Length of beacon data |
| **MAX\_BM\_VARIANT\_NB** | 16 | - | 最多处理几个BM变量  Maximum block mode variants number of one beacon |
| **MAX\_CONNECTED\_PSD\_NB** | 2 | - | Maximum number of PSD ATP can connected at the same time |
| **MAX\_LS\_VARIANT\_NB** | 224 | - | Maximum variants in one line section |
| **MAX\_NTP\_TIME\_ERROR** | 5 | second | 最大允许NTP时钟误差  Maximum allowable error between RTC clock and the NTP time |
| **MAX\_RADAR\_SPEED** | 166667 | mm/s | Maximum speed for radar measurement |
| **MAX\_RESET\_TIME** | 1152000 | cycle | 最大连续运行周期数  Maximum cycle number for ATP work continuously |
| **MAX\_REVERSE\_LIMIT\_STEP** | 2 | - | 最多可定义的倒车限速数组个数  Maximum segmentation for reverse speed table |
| **MAX\_RM\_CONDITION\_NB** | 7 | - | 最多可定义的RM级数 |
| **MAX\_ROLLBACK\_LIMIT\_STEP** | 2 | - | 最多可定义的回溜限速数组个数  Maximum segmentation for rollback speed table |
| **MAX\_SINGULARITY\_NB** | 10000 | - | Maximum number of allowable singularity for one project |
| **MAX\_TIMER\_IMPULSE\_NB** | 1001 | - | Maximum number of allowable safety timer impulses |
| **MAX\_TRACKTION\_MAX**  **\_ACCELERATION\_STEP** | 16 | - | 最多可定义的最大牵引加速度数组个数  Maximum segmentation for traction-acceleration table |
| **MAX\_VITAL\_INPUT\_NB** | 64 |  | Maximum vital input ports |
| **MAX\_VITAL\_OUTPUT\_NB** | 10 |  | Maximum vital output ports |
| **MEM\_LOCATION\_VERSION** | 1 | - | 记忆定位的版本号 |
| **MIN\_TIMER\_IMPULSE\_NB** | 999 | - | Minimum number of allowable safety timer impulses |
| **None** | 0 | - | 即无效值  zero |
| **NUMERIC\_32\_MAX** | 2^30-1 | - | 最大有符号数  Maximum 32bit signed numeric number |
| **NUMERIC\_32\_MIN** | -(2^30) | - | 最小有符号数  Minimum 32bit signed numeric number |
| **ONE\_METER\_TO\_MILLIMETRE** | 1000 | - | 米到毫米的转换  Constant for convertion from meter to millimeter |
| **ONE\_MILLIMETRE\_TO\_MIKRON** | 1000 | - | 毫米到微米的转换  Constant for convertion from millimeter to mikron |
| **OTHER\_ATP\_VALIDITY\_TIME** | 8 | cycle | 冗余CC消息的有效期(周期数)  Validity time for the redundant ATP message |
| **POWER\_OFF** | 1 | - | Disable the sensor |
| **POWER\_ON** | 0 | - | Enable the sensor |
| **REPORT\_AGE\_MAX** | **NUMERIC\_32\_MAX**  ÷4×3 |  | 初始化消息时间  Initialization validity age |
| **SENSOR\_TEST\_START\_TIME** | **ATP\_INTERRUPT\_NB** | - | Interrupt number for waiting before start sensor test |
| **SYNCHRODATE\_TIME\_UNIT** | 0.336 | second | LC/ZC周期时间  LC/[ZC](#ZC) cycle time |
| **SYNCHRODATE\_TIME\_UNIT\_MS** | 336 | millisecond | LC/ZC周期时间（毫秒）  LC/[ZC](#ZC) cycle time |
| **T\_LOCK\_ODOMETER** | 49 | - | 锁存VPB信息的中断号  Interrupt number to lock VPB information |
|  |  |  |  |
| **VIOM\_VALIDITY\_TIME** | 6 | cycle | VIOM消息的有效期(周期数)  Validity cycle's number of VIOM message |
|  |  |  |  |

## Logical Types Definition

本部分用于定义在软件需求文档中用到的各种变量或结构类型。

This part defines variables and structures types used in the description of ATP software requirements.

### Fundamental types

|  |  |  |
| --- | --- | --- |
| Name | Value |  |
| BOOLEAN | **False** |  |
|  | **True** |  |

|  |  |  |
| --- | --- | --- |
| Name | Value |  |
| NUMERIC\_32 | **NUMERIC\_32\_MIN** ~ **NUMERIC\_32\_MAX** |  |

### Enumerated types

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_CALIBRATION\_STATE | **CALI\_WAITING** |
|  | **CALI\_MEASURING** |
|  | **CALI\_VALIDATING** |
|  | **CALI\_COMPLETED** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_CAB\_ID | **END\_UNKOWN** |
|  | **END\_1** |
|  | **END\_2** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_DOT | **UP** |
|  | **DOWN** |
|  | **BOTH** |
|  | **DOT\_UNKOWN** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_EOA\_ZC\_TYPE | **EOA\_NULL\_TYPE\_0** |
|  | **EOA\_NULL\_TYPE\_1** |
|  | **EOA\_END\_OF\_VISIBILITY** |
|  | **EOA\_EXTREMITY\_OF\_AP** |
|  | **EOA\_UNCONTROLLED\_POINT** |
|  | **EOA\_FRONTIER** |
|  | **EOA\_TD\_DISCONTINUITY** |
|  | **EOA\_LIMIT\_OF\_BUFFER\_ZONE** |
|  | **EOA\_CHAINING\_BREAK** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_LOCALIZATION\_STATE | **NOT\_LOCALIZED** |
|  | **MOVING\_INIT** |
|  | **LOCALIZED\_NOT\_CONFIRMED** |
|  | **LOCALIZED\_CONFIRMED** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_MONITORING\_MODE | **OTHER** |
|  | **RMF** |
|  | **RMR** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_ODOMETER\_STATE | **NOT\_INITIALIZED** |
|  | **WAITING\_COG\_POSITION\_CODE\_READY** |
|  | **INITIALIZED** |
|  | **INVALID** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_PSD\_STATE | **PSD\_STATE\_CLOSED** |
|  | **PSD\_STATE\_UNKNOWN** |
|  | **PSD\_STATE\_OPENED** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_SAFE\_IMMO\_BEHAVIOUR | **IB\_NO\_BRAKING\_ACTION** |
|  | **IB\_APPLY\_PARKING\_BRAKE** |
|  | **IB\_APPLY\_EMERGENCY\_BRAKE** |
|  | **IB\_APPLY\_EMERGENCY\_BRAKE\_WHEN\_TRIGGERED** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_SENSOR\_STATUS | **SENSOR\_UNKOWN** |
|  | **SENSOR\_BLOCKED** |
|  | **SENSOR\_CONDUCT** |
|  | **SENSOR\_WRONG** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_SENSOR\_TEST\_RESULT | **TEST\_INCONSISTENT** |
|  | **TEST\_STOPPING** |
|  | **TEST\_IMMOBILE** |
|  | **TEST\_FLOATING** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_SINGULARITY\_TYPE | **SGL\_NONE** |
|  | **SGL\_BMCP** |
|  | **SGL\_CLOSE\_TRACK\_END** |
|  | **SGL\_HAZAR\_EVAC\_ZONE** |
|  | **SGL\_NEW\_BLOCK** |
|  | **SGL\_OPEN\_TRACK\_END** |
|  | **SGL\_OVERLAP\_END** |
|  | **SGL\_OVERLAP\_RELEASE\_ZONE** |
|  | **SGL\_PROTECTION\_ZONE** |
|  | **SGL\_PSD\_ZONE** |
|  | **SGL\_PSR** |
|  | **SGL\_SIGNAL** |
|  | **SGL\_SLEEPING\_ZONE** |
|  | **SGL\_SMI\_ZONE** |
|  | **SGL\_VITAL\_ZONE** |
|  | **SGL\_VPEZ** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_SLIDING\_STATE | **COASTING** |
|  | **BRAKING** |
|  | **SLIDING** |
|  | **SKIDING** |

|  |  |
| --- | --- |
| Name | Value |
| ENUM\_SLIPPING\_STATE | **COASTING** |
|  | **MOTORING** |
|  | **SLIPPING** |
|  | **SKIDING** |

### Aggregated types

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_BM\_VARIANT | |  |  |
|  | LineSection | NUMERIC\_32 | 该变量所在Line Section号 |
|  | Index | NUMERIC\_32 | 该变量在Line Section里的序号 |
|  | Status | BOOLEAN | 该变量状态 |

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_LOCATION\_TRAIN | |  |  |
|  | Uncertainty | NUMERIC\_32 | 定位误差 |
|  | Ext2 | ST\_LOCATION\_UNIT | End2端外侧定位 |
|  | Int2 | ST\_LOCATION\_UNIT | End2端内侧定位 |
|  | Ext2 | ST\_LOCATION\_UNIT | End1端外侧定位 |
|  | Int2 | ST\_LOCATION\_UNIT | End1端内侧定位 |

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_LOCATION\_UNIT | |  |  |
|  | Block | NUMERIC\_32 | 所在BLOCK号 |
|  | Abscissa | NUMERIC\_32 | 相对该BLOCK上行起点的坐标 |
|  | Ort | ENUM\_DOT | 该位置对应的方向 |

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_PSD\_MANAGE | |  |  |
|  | Id | NUMERIC\_32 | PSD的标识 |
|  | Order | NUMERIC\_32 | PSD控制命令字 |

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_PSD\_OPERATION | |  |  |
|  | Id | NUMERIC\_32 | PSD的标识 |
|  | OpeningOrder | BOOLEAN | 是否发开PSD命令 |
|  | ClosingOrder | BOOLEAN | 是否发关PSD命令 |

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | | Logical Type | Description |
| ST\_TSR\_BLOCK | |  |  |
|  | .AbsentOnBlock | BOOLEAN | 该BLOCK是否应用了TSR |
|  | .Speed | NUMERIC\_32 | TSR限速值 |
|  | .MinAbscissa | NUMERIC\_32 | TSR在该BLOCK的开始 |
|  | .MaxAbscissa | NUMERIC\_32 | TSR在该BLOCK的结束坐标 |

## Offline generated codes

考虑到项目实施的灵活性，部分与输入输出相关功能的代码，由离线工具根据项目安全配置数据生成，由在线软件在初始化时读取使用。Table 6‑3为离线代码接口列表，详见[REF11]。

Table 6‑3 Get combined inputs from offline generated codes

|  |  |  |
| --- | --- | --- |
| Offline functions | | Description |
| Offline generated codes for VIOM inputs | | |
|  | GetAllTrainDoorsClosedAndLocked | 两侧车门关闭并锁闭。 |
|  | GetATCcontrolledTrain | CC未被旁路，可执行控车功能 |
|  | GetBlockModeUsed | BM驾驶模式选择，使用BM下的EOA |
|  | GetBMoverlapReleasableSendable | 可通过无线向CI发送Overlap解锁消息 |
|  | GetBMvariantValidWhileTemporallyValid | 使用BM下的变量 |
|  | GetCoercedPermissive[1..4] | 强制允许条件1~4 |
|  | GetConditionForRMlimitSpeed[1..7] | 支持7种不同的RM限速 |
|  | GetCoupledByEnd1 | 驾驶室1端连挂 |
|  | GetCoupledByEnd2 | 驾驶室2端连挂 |
|  | GetDriverInCab\_1 | 司机选择驾驶室1 |
|  | GetDriverInCab\_2 | 司机选择驾驶室2 |
|  | GetEmergencyHandleNotPulledEnd1 | 驾驶室1端的疏散门未被拉下 |
|  | GetEmergencyHandleNotPulledEnd2 | 驾驶室2端的疏散门未被拉下 |
|  | GetEmergencyHandleNotPulledSide | 列车两侧的紧急开门手柄未被拉下 |
|  | GetInhibitControlPSDstatus | 禁止监控PSD状态 |
|  | GetInhibitControlTrainDoorsStatus | 禁止监控车门状态 |
|  | GetInhibitProtectionEvacuationInDistance | 禁止离站时的逃生监控 |
|  | GetInhibitProtectionEvacuationWithStop | 禁止站间停车时的逃生监控 |
|  | GetInhibitProtectionMovingWithoutTDCL | 禁止非开门受权时TDCL丢失监控 |
|  | GetInhibitPSDopeningSupervisedByATP | 禁止PSD开门命令时输出PB |
|  |  |  |
|  | GetMemorizedLocationAuthorized | 授权使用记忆定位 |
|  | GetMotionProtectionInhibition | 禁止CC进行列车位置监控（仅监控RM限速） |
|  | GetNotCoercedRestrictive[1..4] | 非强制限制条件1~4 |
|  | GetRMRselectedDrivingMode | 选择倒车模式 |
|  | GetRouteSetNotNeededSendable | 授权可以给ZC发送RSNN信息 |
|  | GetSignalOverrideSendable | 授权可以给ZC发送关闭信号机信息 |
|  | GetTrainEmergencyBrakeApplied | 紧急制动已施加 |
|  | GetTrainNotCoupled | 列车未连挂 |
|  | GetTrainParkingBrakeApplied | 停车制动已施加 |
|  | GetTrainUnitIntegrity | 列车完整性可以保证 |
|  | GetTSRcontrolInhibition | 禁止TSR监控 |
| Offline generated codes for combined logical inputs | | |
|  | GetLongDistanceReverseAuthorized | 授权长距离倒车模式 |
|  | GetNoUndetectableDanger\_1 | 驾驶室1端没有“未检测到的危险” |
|  | GetNoUndetectableDanger\_2 | 驾驶室2端没有“未检测到的危险” |
| Offline generated codes for VIOM outputs: | | |
|  | GetBooleanForZC(1...16) | Configurable booleans for [ZC](#ZC) |
|  | GetVIOM1VitalOut(1…10) | Vital outputs to VIOM1 for each ports |
|  | GetVIOM2VitalOut(1…10) | Vital outputs to VIOM2 for each ports |

## Message

消息处理机制

Message.Exists, ATP应判断本周期是否从CCNV收到消息。

ATP shall determine whether receive a correct message from CCNV.

Message.VitalChecksumValid，当ATP获取到安全消息时，应当计算进行CRC校验并计算消息中的安全校核字。如果安全消息的CRC或者安全校核字校验错误，则应丢弃该条消息，返回为假；否则，将判断收到正确的安全消息，返回为真。

When ATP received a safety message, shall check the CRC and calculate the check words of the message. If either the CRC or the vital checkword failed, ATP shall discard the message and return **False**; Otherwise, ATP shall consider the message is correct and return **True**.

Message. ReplyLocalCC，判断该消息中的MessageLoopHour是否与本CC一致

def [ReplyLocalCC](#ReplyLocalCC)(MessageLoopHour):

if ((MessageLoopHour >= **CC1\_INIT\_TIME**

and MessageLoopHour <= **CC1\_MAX\_TIME**

and [CoreId](#CoreId) is **END\_1**)

or (MessageLoopHour >= **CC2\_INIT\_TIME**

and MessageLoopHour <= **CC2\_MAX\_TIME**

and [CoreId](#CoreId) is **END\_2**)):

return **True**

else:

return **False**

Message. ReplyDistantCC，判断该消息中的MessageLoopHour是否与远端CC一致

def [ReplyDistantCC](#ReplyDistantCC)(MessageLoopHour):

if ((MessageLoopHour >= **CC1\_INIT\_TIME**

and MessageLoopHour <= **CC1\_MAX\_TIME**

and [OtherCoreId](#OtherCoreId) is **END\_1**)

or (MessageLoopHour >= **CC2\_INIT\_TIME**

and MessageLoopHour <= **CC2\_MAX\_TIME**

and [OtherCoreId](#OtherCoreId) is **END\_2**)):

return **True**

else:

return **False**

Message. ModularSub，loop hour之间的模减

* 考虑到判断消息可用的算法，此处只需强制为严格大于才返回**True**
* 因为在计算[LastAge](#LastAge)时是要+1的，若之前取等号,则加1后可能该消息已无效

def [ModularSub](#ModularSub)(lh\_1, lh\_2):

if (([ReplyLocalCC](#ReplyLocalCC)(lh\_1) and [ReplyLocalCC](#ReplyLocalCC)(lh\_2))

or ([ReplyDistantCC](#ReplyDistantCC)(lh\_1) and [ReplyDistantCC](#ReplyDistantCC)(lh\_2))):

if (lh\_1 - lh\_2 < 0):

return ((lh\_1 - lh\_2) + MAX\_ATP\_LOOP\_HOUR)

else:

return (lh\_1 - lh\_2)

else:

raise LoopHourException

Message.IsMoreRecent，若两端都对应CC1或CC2，且lh\_1 > lh\_2，则返回**True**；否则返回False

def IsMoreRecent(lh\_1, lh\_2):

if (([ReplyLocalCC](#ReplyLocalCC)(lh\_1) and [ReplyLocalCC](#ReplyLocalCC)(lh\_2))

or [ReplyDistantCC](#ReplyDistantCC)(lh\_1) and [ReplyDistantCC](#ReplyDistantCC)(lh\_2)):

return ([ModularSub](#ModularSub)(lh\_1, lh\_2) > 0

and [ModularSub](#ModularSub)(lh\_1, lh\_2) < **ATP\_MESSAGE\_MAX\_DELAY**)

else:

raise LoopHourException

**NOTES：**

**ATP\_MESSAGE\_MAX\_DELAY**是在计算loop hour“模减”时使用的一个比较常数，它的取值足够大，但又远小于**MAX\_ATP\_LOOP\_HOUR**。系统假定，在计算模减时，收到消息中的loop hour和本地维护的时间差值不会大于**ATP\_MESSAGE\_MAX\_DELAY**。

**ATP\_MESSAGE\_MAX\_DELAY** is a constant for the modular subtraction, which shall set as big enough and much less than the **MAX\_ATP\_LOOP\_HOUR**. There is an assumption that ATP do not need to consider the difference between two loop hour is more than or equal to **ATP\_MESSAGE\_MAX\_DELAY**, when calculate the modular minus.

Message.Received，收到消息并且校验正确

def Received(MessageContent, k):

return ([Exists](#Exists)(MessageContent, k)

and [VitalChecksumValid](#VitalChecksumValid)(MessageContent))

Message.Available，判断消息是否可用，即收到消息校验正确，且在有效期内，且比之前收到的更新

* 当收到一条[ATPtime](#ATPtime)-MessageLoopHour恰好等于MessageValidityTime的消息不应认为有效
* 否则计算本周期[LastAge](#LastAge)就将大于消息有效期

def [Available](#Available)(MessageReceived, MessageLoopHour, MessageValidityTime, LastMessageAge, k):

if (MessageReceived

and (([ReplyLocalCC](#ReplyLocalCC)(MessageLoopHour)

and IsMoreRecent([ATPtime](#ATPtime)(k), MessageLoopHour)

and IsMoreRecent(MessageLoopHour + MessageValidityTime, [ATPtime](#ATPtime)(k))

and ([ModularSub](#ModularSub)([ATPtime](#ATPtime)(k) - MessageLoopHour) < LastMessageAge))

or ([ReplyDistantCC](#ReplyDistantCC)(MessageLoopHour)

and IsMoreRecent([OtherATPminTime](#OtherATPminTime)(k), MessageLoopHour)

and IsMoreRecent(MessageLoopHour + MessageValidityTime, [OtherATPmaxTime](#OtherATPmaxTime)(k))

and ([ModularSub](#ModularSub)([OtherATPmaxTime](#OtherATPmaxTime)(k) - MessageLoopHour) < LastMessageAge)))):

return **True**

else:

return **False**

Message.LastAge，当前消息已存活的时间。Age的最大值为**REPORT\_AGE\_MAX**。

def [LastAge](#LastAge)(MessageAvailable, MessageLoopHour, PreviousLastAge, k):

if (Initialization

or PreviousLastAge >= **REPORT\_AGE\_MAX**):

return **REPORT\_AGE\_MAX**

elif (MessageAvailable,

and [ReplyLocalCC](#ReplyLocalCC)(MessageLoopHour)):

return (1 + [ModularSub](#ModularSub)([ATPtime](#ATPtime)(k), MessageLoopHour))

elif (MessageAvailable,

and [ReplyDistantCC](#ReplyDistantCC)(MessageLoopHour)):

return (1 + [ModularSub](#ModularSub)([OtherATPminTime](#OtherATPminTime)(k), MessageLoopHour))

else:

return (1 + PreviousLastAge)

Message. Valid，当前消息是否依然有效

def [Valid](#Valid)(MessageLoopHour, MessageValidityTime, k):

if ([ReplyLocalCC](#ReplyLocalCC)(MessageLoopHour)):

return (([ATPtime](#ATPtime)(k) - MessageLoopHour) <= MessageValidityTime)

elif ([ReplyDistantCC](#ReplyDistantCC)(MessageLoopHour)):

return (([OtherATPmaxTime](#OtherATPmaxTime)(k) - MessageLoopHour) <= MessageValidityTime)

else:

return **False**

## Track Map

需求中用到的线路地图相关方法定义如下：

### Block related

TrackMap.Blocks，获取指定blockId的block。

* 输入：blockId，指定的block标识；
* 返回：block结构

TrackMap.BlockOrigin，返回指定方向上该位置的block起始点的位置

* 输入：loc，指定定位（包括方向）
* 返回；定位，block起点所在定位

TrackMap.NextBlock，获取指定block，指定ort，当前变量下的下个block

* 输入：block，指定block
* 输入：ort，指定下游方向
* 输入：k，当前周期
* 返回：结构体，block

### Singularity related

TrackMap.AllSingsBtwTwoLocs，获取指定范围内的指定类型奇点列表

* 输入：SingType，指定奇点类型
* 输入：Loc1，范围边界定位（包括其方向）
* 输入：Loc2，范围边界定位
* 返回：结构体，指定类型奇点列表

TrackMap.AllSingsInZone，获取指定区域内的指定类型奇点列表

* 输入：SingType，指定奇点类型
* 输入：Loc1，范围边界定位（包括其方向）
* 输入：Length，区域长度
* 返回：结构体，指定类型奇点列表

TrackMap. AllSwitchesInZone，获取指定区域内的道岔列表

* 输入：loc，区域起始边界位置（包括方向）；
* 输入：length，区域长度。
* 返回：结构体，道岔列表

TrackMap.ExistSingBtwTwoLocs，查找从loc1按照指定方向两个位置围成的指定区域内是否存在指定类型的奇点，若有多个奇点，则返回最新的

* 输入：SingType，指定奇点
* 输入：loc1，范围边界定位（包括其指定方向）
* 输入：loc2，范围边界定位
* 返回：结构体，奇点对应的数组元素？

TrackMap.ExistSingularityInZone，查找一个指定区域内是否存在指定类型的奇点

* 输入：SingType，指定类型奇点
* 输入：loc，区域起始坐标（包括区域方向）
* 输入：length，区域长度
* 返回：结构体，奇点对应的数组元素

TrackMap.ExistSingularityInReverseZone，查找指定类型奇点，是否在一个给定位置的反向区域内

* 输入：SingType，指定类型奇点
* 输入：loc，区域结束坐标（包括区域方向，该方向与搜索方向相反）
* 输入：length，区域长度
* 返回：结构体，奇点对应的数组元素

TrackMap.ExistSwitchBtwTwoLocs，loc1和loc2按照ort方向围成的区域内是否存在有道岔，包括发散或汇聚节点

* 输入：loc1，范围边界定位（包括其指定方向）
* 输入：loc2，范围边界定位
* 返回：布尔量

TrackMap.ExistThePole，判断两个Block之间是否存在灯泡线极点。

* 输入Block1，Block1的id
* 输入Block2，Block2的id
* 返回：布尔量。

TrackMap.ExistZoneLocationIncluded，判断指定坐标是否在指定区域内

* 输入：ZoneType，区域类型
* 输入：Location，指定区域位置（包括方向）
* 返回：结构体，区域奇点所指向的相应数组元素

### Beacon related

TrackMap.AreNeighbouredBeacons，表示A信标和B信标是否在线路地图中“相邻”。

* 输入：BeaconA，信标A标识
* 输入：BeaconB，信标B标识
* 返回：布尔量

**NOTES：**

由于设计所限，ATP仅将

对于临近道岔两侧的信标，如果在跨道岔定反位有多个Next\_beacon（），则

如果两个信标之间有道岔，或者灯泡线的“极点”，则不认为这两个信标“相邻”，即不能作为校准信标进行齿距校准，对于非极性车也不能通过这两个信标进行定位初始化。



Figure 6‑1 Neighboured beacon principle in ATP

TrackMap.Beacons，根据信标id获取信标结构体

* 输入：BeaconId，信标标识
* 返回：结构体，信标

TrackMap.BeaconBelongsToCalibrationCouple，判断是否MTIB

* 输入：BeaconId，信标标识
* 返回：布尔量

TrackMap.BmBeaconDirection，BM信标方向

* 输入：BeaconId，信标标识
* 返回：枚举型，运营方向

TrackMap.CalibrationCoupleMaxDistance，两信标间的最大间距

* 输入：BeaconA，信标A标识
* 输入：BeaconB，信标B标识
* 返回：数值型，有范围？

TrackMap.CalibrationCoupleMinDistance，两信标间的最小间距

* 输入：BeaconA，信标A标识
* 输入：BeaconB，信标B标识
* 返回：两信标距离

TrackMap.IsBmBeacon，判断是否BM信标

* 输入：BeaconId，信标标识
* 返回：布尔量

TrackMap.IsCalibrationValidationBeacon，判断A和B是否MTIB和其后的验证信标

* 输入：BeaconA，信标A标识
* 输入：BeaconB，信标B标识
* 返回：布尔量

TrackMap.OrientationOfNeighbouredBeacons，从A到B的方向（需能处理两信标之间有灯泡线“极点”的情况）

* 输入：BeaconA，信标A标识
* 输入：BeaconB，信标B标识
* 返回：枚举型，运营方向

### CBI related

TrackMap.CbiId，根据block索引CBI的id

* 输入：blockId，信标标识
* 返回：数值型，有范围0~32

TrackMap.NumberOfVariants，每个CBI所带的OverlapReleasableZone变量个数

* 输入：CbiId，联锁标识
* 返回：数值型，有范围0~128

### ZC related

TrackMap.ZC，根据id获取ZC

* 输入：ZcId，ZC标识
* 返回：结构体，ZC

TrackMap.ZCId，指定block所在的ZC标识

* 输入：BlockId，block标识
* 返回：数值型，有范围

### Location relationship

TrackMap.CalculateZoneBorder，根据一个区域边界及其长度和方向，计算该区域的另一端边界

* 输入：Loc，原始坐标（包括block和abscissa）
* 输入：Length，非负数，长度
* 返回：根据输入Loc增加Length距离后的位置

TrackMap.DistanceBtwTwoLocs，计算两个位置间的距离，若不可达则报异常

* 输入：Loc1，指定位置1（包括从Loc1指向Loc2的方向）
* 输入：Loc2，指定位置2
* 输入：MaxDistance，最大距离
* 返回：数值型，有范围

TrackMap.IntersectionOfTwoZones，判断区域1和区域2之间有无交集

* 输入：Zone1Loc1，区域1起始位置；
* 输入：Zone1Loc2，区域1结束位置；
* 输入：Zone2Loc1，区域2起始位置；
* 输入：Zone2Loc2，区域2结束位置。
* 返回：若有交集则返回交集部分区域结构体（NewLoc1，NewLoc2）；若无交集返回None

TrackMap.LocationBtwTwoLocs，指定位置是否在指定范围内

* 输入：Loc，指定位置
* 输入：Loc1，范围边界1
* 输入：Loc2，范围边界2
* 返回：布尔型

TrackMap.LocationInZone，指定位置是否在指定区域内

* 输入：Loc，指定位置
* 输入：ZoneLoc，区域起始位置
* 输入：ZoneLen，区域长度
* 返回：布尔型

TrackMap.LocationUpdateExt2，根据输入的位移，更新**END\_2**外侧定位

* 输入：TowardEnd2，列车向END2运行为**True**，否则为**False**
* 输入：PrevEnd2Orient，end2朝向，**UP**或**DOWN**
* 输入：PrevExt2Loc，初始定位（block, abscissa）
* 输入：MaxMotion，最大位移
* 输入：MinMotion，最小位移
* 返回：Ext2新定位，如果没有经过灯泡线“极点”则新定位的方向等同于PrevEnd2Orient，否则变为相反方向

**NOTES：**

在计算**END\_2**的外侧定位Ext2时，应当根据运营方向和列车位移方向，选择使用最大还是最小位移进行。规则如下：

* 若**END\_2**面向**UP**，而列车也向**END\_2**方向运行，即Ext2是车头最大定位，则Ext2的坐标应当在初始定位基础上“增加”最大位移的绝对值；
* 若**END\_2**面向**UP**，而列车向**END\_1**方向运行，即Ext2是车尾最小定位，则Ext2的坐标应当在初始定位基础上“减少”最小位移的绝对值；
* 若**END\_2**面向**DOWN**，而列车也向**END\_2**方向运行，即Ext2是车头最大定位，则Ext2的坐标应当在初始定位基础上“减少”最大位移的绝对值；
* 若**END\_2**面向**DOWN**，而列车向**END\_1**方向运行，即Ext2是车尾最小定位，则Ext2的坐标应当在初始定位基础上“增加”最小位移的绝对值。

TrackMap.OppositeOrientation，获得给定运营方向的反方向

* 输入：Ort，指定运营方向
* 返回：运营方向的反方向

TrackMap.ReachableBetweenTwoLocations，判断两个定位之间是否按指定方向“可达”：即从Loc1搜索，能否在MaxDist内找到Loc2

* 输入：Loc1，起始位置
* 输入：Loc2，终止位置
* 输入：MaxDist，最大距离
* 返回：布尔量

## Assessment on Compliance with EN50128

Compliance of software development with techniques and measures recommended by CENELEC standard EN 50128(2001) is established through correspondence with tables of EN50128(2001) Annex A.

The names of the tables, the references and the conventions are those of the CENELEC standard. A selection of techniques and measure for software development in CC subsystem is given as the following choices:

* √’ – means this technique or measure has been applied.
* ‘-’ – means this technique or measure has not been chosen.
* ‘NA’ – means this technique or measure is not applicable.

Table A.2 Software Requirements Specification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TECHNIQUE/MEASURE | Ref | SIL4 | Remark | Evidence |
| 1. Formal methods including for example CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z and B | B.30 | HR | - | NA |
| 2. Semi-formal methods | D.7 | HR | √ | §4.9.2; 4.10.2; 5.12.3; 5.13.3; 5.17.3; 5.21.3; 5.27.3; 5.34.3; 5.36.3; 5.46.3 |
| 3. Structured methods including for example JSD, MASCOT, SADT, SDL, SSADM, and Yourdon. | B.60 | HR | √ | §5.3; 5.9; 5.18; 5.22; 5.29; 5.35; 5.40 |
| Requirements  1. The Software Requirements Specification will always require a description of the problem in natural language and any necessary mathematical notation that reflects the application.  2. The table reflects additional requirements for defining the specification clearly and precisely. One or more of these techniques shall be selected to satisfy the Software Safety Integrity Level being used. | | | | |

Table A.18 Semi-Formal Methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TECHNIQUE/MEASURE | Ref | SW  SIL4 | Remark | Evidence |
| 1. Logic/Function Block Diags | - | HR | √ | §5.3; 5.9; 5.18; 5.22; 5.29; 5.35; 5.40 |
| 2. Sequence Diagrams | - | HR | √ | §4.9.2, 4.10.2, 5.46.3 |
| 3. Data flow Diagrams | B.12 | R | - | NA |
| 4. Finite State Machines /State Transition Diagrams | B.29 | HR | √ | §5.12.3, 5.13.3, 5.17.3,  5.21.3, |
| 5. Time Petri Nets | B.64 | HR | - | NA |
| 6. Decision/Truth Tables | B.14 | HR | √ | §5.27.3, 5.34.3, 5.36.3 |
| Requirement  A suitable set of techniques shall be chosen according to the software safety integrity level. | | | | |