# 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 51所示模块。
According to the level of the ATP functionality, the software requirements divided into several functional blocks, as shown in Table 51.
Table 51 Functions table

### 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 61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 61, 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 51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，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 51, the basic constitution unit of track map is block, which managed by ZC; on the block, a series of singularities 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 51 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 52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 52. 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 52 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 53所示。
Some of the singularities, there status will change dynamically, such as switch positions, permissive or restrictive signals.
Table 53 Index of variant

### 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.CCTrainType信息，生成TrainType。
On initialization, ATP generates TrainType according to DataPlugContent.CCTrainType from the CC data plug.
‘’’

def TrainType(k):
  
 return 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.CCCoreId信息，生成CoreId。
On initialization, ATP generates CoreId according to DataPlugContent.CCCoreId read from the CC data plug.

def CoreId(k):
  
 return 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(k):
  
 if (CoreId(k) is END\_1):
  
 return END\_2
  
 elif (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.CC\_SSID信息，生成SubSystemId。
On initialization, ATP generates SubSystemId according to DataPlugContent.CC\_SSID from the CC data plug.

def SubSystemId(k):
  
 return DataPlugContent.CC\_SSID

#Category=Functional
#Contribution=SIL4
#Allocation=ATP Software
#Source=[iTC\_CC-SyAD-0210], [iTC\_CC-SyAD-0371]
[End]

Check the project settingATP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，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 61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, 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 according to the Table 61;
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，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 and detect the track map data with VCP which was generated by CC offline tools. If the data is correct, ATP shall generate 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 63所示，详见[REF11]。
Some parts of codes in DATA.VES, related to input and output functions, as project configuration data, ATP links them during initialization and generates Offline structure as Table 63, 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 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，来自data plug的列车类型；
CoreId，来自data plug的所在车头标识；
SubSystemId，来自data plug的列车标识；
TrackMap，线路地图中相应每个ZC区的版本号；
ATPsetting，配置数据的版本号；
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. If the information is as same, ATP shall set IdenticalVersionOfDualCPU as True; otherwise, it will set IdenticalVersionOfDualCPU as False.
TrainType, the train type from data plug;
CoreId, the train end identification from data plug;
SubSystemId, the train identification from data plug;
TrackMap, the version number of each ZC region in line map;
ATPsetting, the version number for the configuration data;
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 information. If TrainKnown considered as False, ATP shall set all output ports as restricted.

def TrainKnown(k):
  
 return ((TrainType(k) == ATPsetting.TrainTypeId)
  
 and (CoreId(k) is END\_1
  
 or CoreId(k) is END\_2)
  
 and 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
Through the communication with CCNV, ATP judges NonVitalRequestreceived from CCNV and generates NonVitalRequestReady If received a new message.

def NonVitalRequestReady(k):
  
 return Message.Exists(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 stands for the effectiveness of CCNV message. If there is no updating CCNV message past the CCNV\_VALIDITY\_CYCLES, ATOcontrolTimeValid is set as False.

def ATOcontrolTimeValid(k):
  
 if (NonVitalRequestReady(k)):
  
 ATOcontrolTimeValid = True
  
 ATOcontrolTimer = 0
  
 elif (ATOcontrolTimer(k-1) &lt; CCNV\_VALIDITY\_CYCLES):
  
 ATOcontrolTimer = ATOcontrolTimer(k-1) + 1
  
 else:
  
 ATOcontrolTimeValid = False
  
 return ATOcontrolTimeValid

#Category= Functional
#Contribution=SIL0
#Allocation=ATP Software
#Source=[iTC\_CC-SyAD-1044]
[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| ATOcontrolTimeValid | × | √ | REF BOOLEAN \* MERGEFORMAT BOOLEAN |
| ATPsetting | √ | × | REF \_Ref305075755 \w \h \* MERGEFORMAT [REF11] |
| CoreId | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| Offline | √ | × | REF \_Ref305075755 \w \h \* MERGEFORMAT [REF11] |
| OtherCoreId | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| SafeApplicationVersion | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| SubSystemId | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| TrackMap | √ | × | REF \_Ref305075755 \w \h \* MERGEFORMAT [REF11] |
| TrainType | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |

## F12-Manage Train Status

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPtime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| VIOM1VitalInput | External | REF \_Ref348805131 \h Interface with VIOM |
| VIOM2VitalInput | External | REF \_Ref348805131 \h Interface with VIOM |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| VIOM1VitalInputsAvailable | √ | √ | REF BOOLEAN BOOLEAN |
| VIOM1VitalInputsLastAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| VIOM1VitalInputsReceived[2] | √ | √ | REF BOOLEAN BOOLEAN |
| VIOM1VitalInputsValid | √ | √ | REF BOOLEAN BOOLEAN |
| VIOM2VitalInputsAvailable | √ | √ | REF BOOLEAN BOOLEAN |
| VIOM2VitalInputsLastAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| VIOM2VitalInputsReceived[2] | √ | √ | REF BOOLEAN BOOLEAN |
| VIOM2VitalInputsValid | √ | √ | REF BOOLEAN 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 411，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 411, 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(viomId, k):
  
 return Message.Received(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.

def VIOM1VitalInputsAvailable(ViomId, k):
  
 return Message.Available(VIOM1VitalInputsReceived(ViomId, k),
  
 VIOM1VitalInput(ViomId).AtpLoopHour,
  
 VIOM\_VALIDITY\_TIME,
  
 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(ViomId, k):
  
 return Message.LastAge(VIOM1VitalInputsAvailable(ViomId, k),
  
 VIOM1VitalInput(ViomId).AtpLoopHour,
  
 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.
In initialization, set 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 as the viomLoopHour of the message.
Otherwise, LatestVIOM1LoopHourVIOM keeps unchanged.

def LatestVIOM1LoopHourVIOM(ViomId, k):
  
 if (VIOM1VitalInputsAvailable(k)
  
 or (not VIOM1VitalInputsValid(k-1)
  
 and VIOM1VitalInputsReceived(ViomId, k))):
  
 return VIOM1VitalInput[ViomId].ViomLoopHour
  
 else:
  
 return 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(ViomId, k):
  
 return Message.Received(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.

def VIOM2VitalInputsAvailable(ViomId, k):
  
 return Message.Available(VIOM2VitalInputsReceived(ViomId, k),
  
 VIOM2VitalInput(ViomId).AtpLoopHour,
  
 VIOM\_VALIDITY\_TIME,
  
 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(viomId, k):
  
 return Message.LastAge(VIOM2VitalInputsAvailable(viomId, k),
  
 VIOM2VitalInput(viomId).AtpLoopHour,
  
 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\_2by the term LatestVIOM2LoopHourVIOM.
In initialization, set 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 as the viomLoopHour of the message.
Otherwise, LatestVIOM2LoopHourVIOM keeps unchanged.

def LatestVIOM2LoopHourVIOM(ViomId, k):
  
 if (VIOM2VitalInputsAvailable(k)
  
 or (not VIOM2VitalInputsValid(k-1)
  
 and VIOM2VitalInputsReceived(ViomId, k))):
  
 return VIOM2VitalInput[ViomId].ViomLoopHour
  
 else:
  
 return 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 54所示，供离线生成代码使用，其详细定义见[REF11]。
ATP shall perform logical &quot;or&quot; operation between two VIOM messages at each cab, and generate one identical VIOM input message for each cab, as shown in Table 54. The message as an interface, are available to offline generated code, defined in [REF11].
Table 54 Identical VIOM Inputs for offline application

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

[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(k):
  
 return (Message.Valid(VIOM1VitalInput(1).AtpLoopHour, VIOM\_VALIDITY\_TIME)
  
 or Message.Valid(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(k):
  
 return (Message.Valid(VIOM2VitalInput(1).AtpLoopHour, VIOM\_VALIDITY\_TIME)
  
 or Message.Valid(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 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 of this end is also ineffective.
If only one VIOM message from one train end is invalid, the 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.
#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 54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 54.
Figure 54 Configurable Rolling-stock Inputs
ATP支持的所有可配置输入见Table 55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 55. All these variables are generated by offline tools and as a part of the project data read by ATP during initialization.
Table 55 Configurable Inputs from Rolling Stock

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

Train coupled status managementATP支持两车的连挂，如Figure 55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 55，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 55 Train Coupled Status
除去Figure 55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 or 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(k):
  
 return Offline.GetCoupledByEnd1(k)
  
 def CoupledByEnd2(k):
  
 return Offline.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(k):
  
 return Offline.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(k):
  
 if (TrainNotCoupled(k)
  
 and not TrainCoupledByEnd1(k)
  
 and not TrainCoupledByEnd2(k)):
  
 return TRAIN\_NO\_COUPLED
  
 elif (not TrainNotCoupled(k)
  
 and TrainCoupledByEnd1(k)
  
 and not TrainCoupledByEnd2(k)):
  
 return TRAIN\_COUPLED\_END1
  
 elif (not TrainNotCoupled(k)
  
 and not TrainCoupledByEnd1(k)
  
 and 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，TrainCoupledByEnd1和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与连挂类型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为False；
当满足以下条件后，ATP将计算新的列车类型TrainType并写入Dataplug，同时设置MatchRebootCondition为True，请求重启ATP。
CoupledTypeInconsistent为True；
且TrainCoupledType不是TRAIN\_COUPLED\_UNKNOWN；
且列车处于TrainFilteredStopped状态。
其他情况，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 | √ | √ | REF BOOLEAN BOOLEAN |
| IdenticalInputs | √ | √ | REF BOOLEAN BOOLEAN |
| LatestVIOM1LoopHourVIOM | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| LatestVIOM2LoopHourVIOM | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| MatchRebootCondition | √ | √ | REF BOOLEAN BOOLEAN |
| TrainCoupledType | √ | √ | REF NUMERIC\_32 \h 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 | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| OtherCCsynchroReport | External | REF \_Ref348814088 \h Interface with Distant ATP |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| LastOtherATPmessageAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| OtherATPmessageAvailable | √ | √ | REF BOOLEAN BOOLEAN |
| OtherATPmessageReceived | √ | √ | REF BOOLEAN BOOLEAN |

ProcessingLoop hour managementATP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，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 stands for the ATP loop hour of this train END.
Based on CoreId, ATP initialize 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(k):
  
 if (CoreId(k) is END\_1):
  
 if (Initialization):
  
 return CC1\_INIT\_TIME
  
 elif (ATPtime(k-1) &gt;= CC1\_MAX\_TIME):
  
 return CC1\_INIT\_TIME
  
 else:
  
 return ATPtime(k-1) + 1
  
 else:
  
 if (Initialization):
  
 return CC2\_INIT\_TIME
  
 elif (ATPtime(k-1) &gt;= CC2\_MAX\_TIME):
  
 return CC2\_INIT\_TIME
  
 else:
  
 return 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 ATPATP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(k):
  
 return Message.Received(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, ATP shall judge the effectiveness of message from the redundant ATP, shown as following pseudo-codes:

def OtherATPmessageAvailable(k):
  
 return Message.Available(OtherATPmessageReceived(k),
  
 OtherCCsynchroReport.LatestTimeOtherCore,
  
 OTHER\_ATP\_VALIDITY\_TIME,
  
 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(k):
  
 return Message.LastAge(OtherATPmessageAvailable(k),
  
 OtherCCsynchroReport.LatestTimeOtherCore,
  
 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 represents the effectiveness of the messages from redundant ATP. If this message is invalid, ATP will set OtherATPmessageValid as False; otherwise, it is set as True.

def OtherATPmessageValid(k):
  
 return Message.Valid(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 stands for the local ATP maintained minimum time of the redundant ATP. The setting rule is as following:
In initialization, ATP set the OtherATPminTime as default value based on the 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 as the current time in the message.
Or else:, if the redundant ATP message is still effective, ATP will add 1 in the 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 as in the message.
Otherwise, accumulate OtherATPminTime.

if (CoreId == END\_1)
  
 if (Initialization)
  
 OtherATPminTime = CC2\_INIT\_TIME
  
 elif ((OtherATPmessageAvailable(k) == True)
  
 or ((OtherATPmessageValid(k) == False)
  
 and Exists(OtherCCsynchroReport)))
  
 OtherATPminTime = OtherCCsynchroReport.CurrentTime
  
 else:
  
 if (OtherATPminTime(k-1) &gt;= CC2\_MAX\_TIME)
  
 OtherATPminTime = CC2\_INIT\_TIME
  
 else:
  
 OtherATPminTime = OtherATPminTime(k-1) + 1
  
 else:
  
 if (Initialization)
  
 OtherATPminTime = CC1\_INIT\_TIME
  
 elif ((OtherATPmessageAvailable(k) == True)
  
 or ((OtherATPmessageValid(k) == False)
  
 and Exists(OtherCCsynchroReport)))
  
 OtherATPminTime = OtherCCsynchroReport.CurrentTime
  
 else:
  
 if (OtherATPminTime(k-1) &gt;= CC1\_MAX\_TIME)
  
 OtherATPminTime = CC1\_INIT\_TIME
  
 else:
  
 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 stands for the local ATP maintained maximum time of the redundant ATP. The rules to update OtherATPmaxTime are similar with OtherATPminTime except that when received a new message from the redundant ATP, the OtherATPmaxTime shall add the maximum transmission delay in network.

if (CoreId == END\_1)
  
 if (Initialization)
  
 OtherATPmaxTime = CC2\_INIT\_TIME
  
 elif ((OtherATPmessageAvailable(k) == True)
  
 or ((OtherATPmessageValid(k) == False)
  
 and Exists(OtherCCsynchroReport)))
  
 OtherATPmaxTime = OtherCCsynchroReport.CurrentTime
  
 + LoopHourModularSub(ATPtime(k),
  
 OtherCCsynchroReport.LatestTimeOtherCore)
  
 else:
  
 if (OtherATPmaxTime(k-1) &gt;= CC2\_MAX\_TIME)
  
 OtherATPmaxTime = CC2\_INIT\_TIME
  
 else:
  
 OtherATPmaxTime = OtherATPmaxTime(k-1) + 1
  
 else:
  
 if (Initialization)
  
 OtherATPmaxTime = CC1\_INIT\_TIME
  
 elif ((OtherATPmessageAvailable(k) == True)
  
 or ((OtherATPmessageValid(k) == False)
  
 and Exists(OtherCCsynchroReport)))
  
 OtherATPmaxTime = OtherCCsynchroReport.CurrentTime
  
 + LoopHourModularSub(ATPtime(k),
  
 OtherCCsynchroReport.LatestTimeOtherCore)
  
 else:
  
 if (OtherATPmaxTime(k-1) &gt;= CC1\_MAX\_TIME)
  
 OtherATPmaxTime = CC1\_INIT\_TIME
  
 else:
  
 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(k):
  
 if (Initialization
  
 or (not OtherATPmessageValid(k))):
  
 OtherATP.LatestTimeOtherCore = INVALID\_LOOP\_HOUR
  
 OtherATP.CoreId = None
  
 OtherATP.BeaconId = None
  
 OtherATP.EnableDoorOpening\_A = False
  
 OtherATP.EnableDoorOpening\_B = False
  
 OtherATP.PsdManagerOpeningOrder = False
  
 OtherATP.PsdIdSide\_A = None
  
 OtherATP.PsdValiditySide\_A = None
  
 OtherATP.PsdClosedSide\_A = False
  
 OtherATP.PsdIdSide\_B = None
  
 OtherATP.PsdValiditySide\_B = None
  
 OtherATP.PsdClosedSide\_B = False
  
 OtherATP.ZcVersion = None
  
 OtherATP.LocatedOnKnownPath = False
  
 OtherATP.LocatedWithMemLocation = False
  
 OtherATP.Location.Ext2 = None
  
 OtherATP.Location.Uncertainty = None
  
 OtherATP.Location.Ext1 = None
  
 OtherATP.SleepZoneId = None
  
 OtherATP.SleepZoneVersion = None
  
 OtherATP.MotionSinceLastReloc = None
  
 OtherATP.MotionSinceMemLoc = None
  
 OtherATP.TrainFilteredStopped = False
  
 OtherATP.SafetyParameterVersion = None
  
 OtherATP.SafetyApplicationVersion = None
  
 OtherATP.CC\_SSID = None
  
 OtherATP.OverlapExpired = False
  
 elif (OtherATPmessageAvailable(k)):
  
 OtherATP.LatestTimeOtherCore = OtherCCsynchroReport.LatestTimeOtherCore(k)
  
 OtherATP.CoreId = OtherCCsynchroReport.CoreId
  
 OtherATP.BeaconId = OtherCCsynchroReport.BeaconId
  
 OtherATP.EnableDoorOpening\_A = OtherCCsynchroReport.EnableDoorOpening\_A
  
 OtherATP.EnableDoorOpening\_B = OtherCCsynchroReport.EnableDoorOpening\_B
  
 OtherATP.PsdManagerOpeningOrder = OtherCCsynchroReport.PsdManagerOpeningOrder
  
 OtherATP.PsdIdSide\_A = OtherCCsynchroReport.PsdIdSide\_A
  
 OtherATP.PsdValiditySide\_A = OtherCCsynchroReport.PsdValiditySide\_A
  
 OtherATP.PsdClosedSide\_A = OtherCCsynchroReport.PsdClosedSide\_A
  
 OtherATP.PsdIdSide\_B = OtherCCsynchroReport.PsdIdSide\_B
  
 OtherATP.PsdValiditySide\_B = OtherCCsynchroReport.PsdValiditySide\_B
  
 OtherATP.PsdClosedSide\_B = OtherCCsynchroReport.PsdClosedSide\_B
  
 OtherATP.ZcVersion = OtherCCsynchroReport.ZcVersion
  
 OtherATP.LocatedOnKnownPath = OtherCCsynchroReport.LocatedOnKnownPath
  
 OtherATP.LocatedWithMemLocation = OtherCCsynchroReport.LocatedWithMemLocation
  
 OtherATP.Location.Ext2 = OtherCCsynchroReport.Location.Ext2
  
 OtherATP.Location.Uncertainty = OtherCCsynchroReport.Location.Uncertainty
  
 OtherATP.Location.Ext1 = OtherCCsynchroReport.Location.Ext1
  
 OtherATP.SleepZoneId = OtherCCsynchroReport.SleepZoneId
  
 OtherATP.SleepZoneVersion = OtherCCsynchroReport.SleepZoneVersion
  
 OtherATP.MotionSinceLastReloc = OtherCCsynchroReport.MotionSinceLastReloc
  
 OtherATP.MotionSinceMemLoc = OtherCCsynchroReport.MotionSinceMemLoc
  
 OtherATP.TrainFilteredStopped = OtherCCsynchroReport.TrainFilteredStopped
  
 OtherATP.SafetyParameterVersion = OtherCCsynchroReport.SafetyParameterVersion
  
 OtherATP.SafetyApplicationVersion = OtherCCsynchroReport.SafetyApplicationVersion
  
 OtherATP.CC\_SSID = OtherCCsynchroReport.CC\_SSID
  
 OtherATP.OverlapExpired = OtherCCsynchroReport.OverlapExpired
  
 else:
  
 pass
  
 return 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 | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| OtherATP | √ | √ | REF ST\_SYNCHRO\_REPORT \h ST\_SYNCHRO\_REPORT |
| OtherATPmaxTime | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| OtherATPmessageValid | √ | √ | REF BOOLEAN BOOLEAN |
| OtherATPminTime | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |

## F14-Manage Variants in Block Mode

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPtime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| BeaconMessage | Internal | REF \_Ref345055658 \h F21-Manage Interface with |
| BeaconMessageReceive | Internal | REF \_Ref345055658 \h F21-Manage Interface with |
| CBIvariantReport | Internal | REF \_Ref345055764 \h F41-Determine the EOA |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| OtherATPmaxTime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainFrontOrientation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainLocalized | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainLocatedOnKnownPath | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| BMbeaconVariantValue | √ | × | REF BOOLEAN BOOLEAN |
| BMbeaconVariants | √ | √ | REF BOOLEAN BOOLEAN |
| BMcbiVariantValue | √ | × | REF BOOLEAN BOOLEAN |
| BMcbiVariants | √ | √ | REF BOOLEAN BOOLEAN |
| BMvariantValidLastRisingAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| BeaconVariantsUpdating | √ | √ | REF BOOLEAN BOOLEAN |
| CBIvariantAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| CBIvariantReportAvailable | √ | √ | REF BOOLEAN BOOLEAN |
| CBIvariantReportReceived | √ | √ | REF BOOLEAN BOOLEAN |
| UsedBMbeaconId | √ | √ | REF NUMERIC\_32 \h 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 represents that either of train end chooses BM mode.

def BlockModeUsed(k):
  
 return Offline.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 shows whether it is in the BM mode.

def BMvariantValidWhileTemporallyValid(k):
  
 return Offline.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）；
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 used to determine ATP whether to update the BM variants in this cycle.
If all the following conditions are fulfilled, ATP shall set 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 as False.

def BeaconVariantsUpdating(k):
  
 return (BMvariantValidWhileTemporallyValid(k)
  
 and BeaconMessageReceive(k)
  
 and TrackMap.IsBmBeacon(BeaconMessage.ID)
  
 and not TrainFilteredStopped(k)
  
 and (not TrainLocalized(k-1)
  
 or (TrackMap.BmBeaconDirection(BeaconMessage.ID)== 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)；
或BMbeaconReadAge已大于ATPsetting.VariantsBMfullValidityTime；
或本周期收到的BM信标（BeaconVariantsUpdating为True）中DefaultMessage为True或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(k):
  
 if (Initialization
  
 or not BMvariantValidWhileTemporallyValid(k)
  
 or BMbeaconReadAge(k-1) &gt; (ATPsetting.VariantsBMfullValidityTime - 1)
  
 or (BeaconVariantsUpdating(k)
  
 and (DefaultMessage(k) or not BlockModeVariantAvailable(k)))
  
 or (TrainLocalized(k-1)
  
 and (not TrainLocalized(k)
  
 or TrackMap.BmBeaconDirection(UsedBMbeaconId(k-1))
  
 is not TrainFrontOrientation(k-1)))):
  
 return REPORT\_AGE\_MAX
  
 elif (BeaconVariantsUpdating(k)):
  
 return 1
  
 else:
  
 return 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大于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.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 are ST\_BM\_VARIANT. The rules to update the BM variants are as follows:
if the BMbeaconReadAge is larger than the ATPsetting.VariantsBMfullValidityTime, ATP shall set all BM variants as restricted status.
Else If the BeaconVariantsUpdating is True, then ATP update the 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(lineSection, VarIndex, k):
  
 if (BMbeaconReadAge(k) &gt; ATPsetting.VariantsBMfullValidityTime):
  
 return False
  
 else:
  
 for Var in range(0, MAX\_BM\_VARIANT\_NB):
  
 if (BMbeaconVariants[Var].LineSection == LineSection
  
 and BMbeaconVariants[Var].Index == VarIndex):
  
 return 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），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 records the used BM variants came from which BM beacon:
When one of the following conditions fulfilled, ATP clear the 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;
Otherwise, keep this value unchanged.

def UsedBMbeaconId(k):
  
 if (BeaconVariantsUpdating(k)):
  
 return BeaconMessage.Id
  
 elif (Initialization
  
 or not BMvariantValidWhileTemporallyValid(k)
  
 or (TrainLocalized(k-1)
  
 and (TrackMap.BmBeaconDirection(UsedBMbeaconId(k-1))
  
 is not TrainFrontOrientation(k-1)))
  
 or (TrainLocalized(k-1) and not TrainLocalized(k))):
  
 return None
  
 else:
  
 return 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：
对于联锁发送的变量与ZC发送的变量，在ATPsetting中定义有对应关系，可以通过联锁变量索引ZC变量。
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(k):
  
 if (not BMvariantValidWhileTemporallyValid(k)):
  
 BMvariantValidLastRisingAge = 0
  
 else:
  
 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(cbi, k):
  
 return Message.Received(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(cbi, k):
  
 return Message.Available(CBIvariantReportReceived(cbi, k),
  
 CBIvariantReport(cbi).CcLoopHour,
  
 ATPsetting.VariantsBMlowValidityTime,
  
 min(CBIvariantReportLastAge(cbi, k-1),
  
 CBIminProductionAge(cbi, k-1),
  
 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(cbi, k):
  
 return Message.LastAge(CBIvariantReportAvailable(cbi, k),
  
 CBIvariantReport(cbi).CcLoopHour,
  
 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(cbi, k):
  
 if (CBIvariantReportAvailable(cbi, k)):
  
 for idx in range(0, CBIvariantReport.NumberOrVariants):
  
 BMcbiVariants[cbi].Variants[idx] = CBIvariantReport.Variant[idx]
  
 else:
  
 pass
  
 return 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进行计算，在判断有效期时导向安全侧。

def CBIvariantAge(cbi, k):
  
 if (Initialization
  
 or CBIvariantAge(k-1) &gt;= REPORT\_AGE\_MAX):
  
 return REPORT\_AGE\_MAX
  
 elif (CBIvariantReportAvailable(cbi, k)
  
 and Message.ReplyLocalCC(CBIvariantReport(cbi).CcLoopHour)):
  
 return (1 + Message.ModularSub(ATPtime(k), CBIvariantReport(cbi).CcLoopHour))
  
 elif (CBIvariantReportAvailable(cbi, k)
  
 and Message.ReplyDistantCC(CBIvariantReport(cbi).CcLoopHour)):
  
 return (1 + Message.ModularSub(OtherATPmaxTime(k), CBIvariantReport(cbi).CcLoopHour))
  
 else:
  
 return (1 + 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(cbi, k):
  
 if (not BlockModeUsed(k)
  
 or not CBIvariantMoreAvailableThanBeacon(cbi, k)
  
 or CBIvariantAge(cbi, k) &lt;= 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(CbiId, VarIndex, k):
  
 if (CBIvariantAge(CbiId, k) &gt; ATPsetting.VariantsBMfullValidityTime):
  
 return False
  
 else:
  
 return 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(cbiId, k):
  
 if (CBIvariantReportAvailable(cbiId, k)):
  
 return CBIvariantReport(cbiId).CcLoopHour
  
 else:
  
 return 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(CbiId, k):
  
 if (Initialization
  
 or not TrainLocatedOnKnownPath(k-1)
  
 or not ATPsetting.BlockModeThroughRadio(k)):
  
 return False
  
 else:
  
 return (UsedBMbeaconId(k) is None
  
 or (CBIvariantAge(CbiId, k) &lt;= ATPsetting.VariantsBMfullValidityTime
  
 and (CBIvariantReportLastAge(CbiId, k)
  
 &lt;= BMbeaconReadAge(k) + ATPsetting.VariantsBMradioPriorityDelay)
  
 and (CBIvariantReportLastAge(CbiId, k) &lt;= 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 &lt;= 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和CBIminProductionAge。
[iTC\_CC\_ATP-SwRS-0628]
BMvariantValue，统一来自BM信标和CBI无线的BM变量

def BMvariantValue(Variant, k):
  
 if (CBIvariantMoreAvailableThanBeacon(Variant.Cbi.Id, k)):
  
 return BMcbivariantValue(Variant.Cbi.Id, Variant.Cbi.Index, k)
  
 else:
  
 return 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(cbi, k):
  
 if (not BMvariantValidWhileTemporallyValid(k)):
  
 return 0
  
 elif (CBIvariantMoreAvailableThanBeacon(cbi, k)):
  
 return max(0, ATPsetting.VariantsBMfullValidityTime - CBIvariantAge(cbi, k))
  
 else:
  
 return max(0, ATPsetting.VariantsBMfullValidityTime - 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 | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| BMbeaconReadAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| BMvariantRemainingTime | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| BMvariantValidWhileTemporallyValid | √ | √ | REF BOOLEAN BOOLEAN |
| BMvariantValue | √ | × | REF BOOLEAN BOOLEAN |
| BlockModeUsed | √ | √ | REF BOOLEAN BOOLEAN |
| CBIvariantMoreAvailableThanBeacon | √ | √ | REF BOOLEAN BOOLEAN |
| CBIvariantReportLastAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |

## F15-Manage Variants in CBTC Mode

管理CBTC模式下的变量

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPtime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| BMvariantValidWhileTemporallyValid | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| BMvariantValue | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| EOA\_Report | External | REF \_Ref348814366 \h Interface with ZC |
| OtherATP | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| OtherATPmaxTime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| OtherATPmessagevalid | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| OtherATPminTime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| SafeApplicationVersion | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| VariantReport | External | REF \_Ref348814366 \h Interface with ZC |
| VersionAuthorization | External | REF \_Ref348814411 \h Interface with LC |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| CBTCvariantValue | √ | × | REF BOOLEAN BOOLEAN |
| EOAReportReceived | √ | √ | REF BOOLEAN BOOLEAN |
| EOAgroundAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| LastEOAReportAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| LastVersionReportAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| ReceivedEOAreport | √ | √ | REF ST\_EOA\_RCV \h ST\_EOA\_RCV |
| ReceivedVariantReport | √ | × | REF ST\_VARIANT\_RCV \h ST\_VARIANT\_RCV |
| ReceivedVersionMessages | √ | √ | REF ST\_VERSION\_RCV \h ST\_VERSION\_RCV |
| SameVersionWithDistantCore | √ | √ | REF BOOLEAN BOOLEAN |
| VariantGroundAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| VariantReportReceived | √ | √ | REF BOOLEAN BOOLEAN |
| VersiionAuthorizationAvailable | √ | √ | REF BOOLEAN BOOLEAN |
| VersiionAuthorizationReceived | √ | √ | REF BOOLEAN BOOLEAN |
| ZCmessageReady | √ | √ | REF BOOLEAN BOOLEAN |

### Processing

Version authorization by LC[iTC\_CC\_ATP-SwRS-0630]
VersionAuthorizationReceived，收到版本授权

def VersionAuthorizationReceived(lcId, k):
  
 return Message.Received(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(VersionAuthorizationReceived(lcId, k),
  
 VersionAuthorization(lcId).CcLoopHour,
  
 ATPsetting.VersionsValidityTime,
  
 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(lcId, k):
  
 return Message.LastAge(VersionAuthorizationAvailable(lcId, k),
  
 VersionAuthorization(lcId).CcLoopHour,
  
 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 | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | 版本授权信息过期时间 | VitalAuthorization[ REF MAX\_ZC\_NB \h MAX\_ZC\_NB] |

def ReceivedVersionMessages(LcId, k):
  
 if (Initialization):
  
 ReceivedVersionMessages = None
  
 elif (VersionAuthorizationAvailable(LcId, k)):
  
 if (Message.ReplyLocalCC(VersionAuthorization(LcId).CcLoopHour)):
  
 NewValidity = (VersionAuthorization(LcId).CcLoopHour
  
 + ATPsetting.VersionsValidityTime)
  
 else:
  
 NewValidity
  
 = (ATPtime(k) + ATPsetting.VersionsValidityTime
  
 - (OtherATPmaxTime(k) - VersionAuthorization(LcId).CcLoopHour))
  
 for ZcId in range(0, MAX\_ZC\_NB):
  
 if (TrackMap.Zc[ZcId].LcId == LcId):
  
 ReceivedVersionMessages[ZcId].VitalAuthorization
  
 = VersionAuthorization(LcId).VitalAuthorization[ZcId]
  
 ReceivedVersionMessages[ZcId].ValidityTime = NewValidity
  
 else:
  
 pass
  
 else:
  
 ReceivedVersionMessages = ReceivedVersionMessages(k-1)
  
 return 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(ZcId, k):
  
 if (Message.IsMoreRecent
  
 (ReceivedVersionMessages(TrackMap.Zc[ZcId].LcId ,k)[zcId].ValidityTime,
  
 ATPtime(k))):
  
 return ReceivedVersionMessages(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(k):
  
 if (OtherATPmessageValid(k)
  
 and (OtherATP.SafetyParameterVersion == ATPsetting.SafetyParameterVersion)
  
 and (OtherATP.SafetyApplicationVersion == SafeApplicationVersion)):
  
 for ZcId in range(0, MAX\_ZC\_NB):
  
 if (OtherATP.ZcVersion[ZcId] != TrackMap.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(k):
  
 return Message.Received(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 represents that an available EOA and variants message from ZC received in this cycle.

def ZCmessageReady(k):
  
 return (Message.Available(EOAReportReceived(k),
  
 EOA\_Report.CcLoopHour,
  
 ATPsetting.EOAvalidityTime,
  
 LastEOAReportAge(k-1),
  
 k)
  
 and (VersionAuthorizedByLC(SSIDofZC, k))
  
 and (Message.ReplyLocalCC(EOA\_Report.CcLoopHour)
  
 or 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 represents the value calculated by current ATP time minus the previous loc-report number and the EOA consuming time in ZC.

def LastEOAReportAge(k):
  
 return Message.LastAge(ZCmessageReady(k),
  
 EOA\_Report.CcLoopHour,
  
 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 stands for the number of CC cycle when receiving the EOA information.

def EOAgroundAge(k):
  
 if (Initialization):
  
 EOAgroundAge.WithoutSpacing = REPORT\_AGE\_MAX
  
 EOAgroundAge.Classic = REPORT\_AGE\_MAX
  
 elif (ZCmessageReady(k)):
  
 EOAgroundAge.WithoutSpacing = (round.ceil
  
 ((EOA\_Report.MessageContainerCreationTime
  
 - EOA\_Report.WithoutSpacingEoaCreationTime)
  
 \* SYNCHRODATE\_TIME\_UNIT\_MS / ATP\_CYCLE\_TIME\_MS))
  
 EOAgroundAge.Classic = round.ceil((EOA\_Report.MessageContainerCreationTime
  
 - EOA\_Report.EoaCreationTime)
  
 \* SYNCHRODATE\_TIME\_UNIT\_MS / ATP\_CYCLE\_TIME\_MS)
  
 else:
  
 EOAgroundAge = EOAgroundAge(k-1)
  
 return 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 |

def ReceivedEOAreport(k):
  
 if (Initialization):
  
 ReceivedEOAreport = None
  
 elif (ZCmessageReady(k)):
  
 ReceivedEOAreport.TrainFrontEnd = EOA\_Report.TrainFrontEnd
  
 ReceivedEOAreport.Classic = UpdateReceivedEoa(EOA\_Report.CcLoopHour,
  
 EOAgroundAge(k).Classic,
  
 EOA\_Report.Classic,
  
 ReceivedEOAreport(k-1).Classic)
  
 ReceivedEOAreport.WithoutSpacing = (UpdateReceivedEoa
  
 (EOA\_Report.CcLoopHour,
  
 EOAgroundAge(k).WithoutSpacing,
  
 EOA\_Report.WithoutSpacing,
  
 ReceivedEOAreport(k-1).WithoutSpacing))
  
 ReceivedEOAreport.CcLoopHour = EOA\_Report.CcLoopHour
  
 else:
  
 ReceivedEOAreport = ReceivedEOAreport(k-1)
  
 if (Message.IsMoreRecent(ATPtime(k), ReceivedEOAreport.Classic.ValidityTime)):
  
 clean\_reseived\_eoa\_classic
  
 if (Message.IsMoreRecent(ATPtime(k), ReceivedEOAreport.WithoutSpacing.ValidityTime)):
  
 clean\_reseived\_eoa\_without\_space
  
 return ReceivedEOAreport

其中UpdateReceivedEoa定义如下:

def UpdateReceivedEoa(NewEoaLoopHour, EoaGroundAge, NewReceivedEoa, PreviousReceivedEoa):
  
 if (Message.ReplyLocalCc(NewEoaLoopHour)):
  
 NewValidity = (NewEoaLoopHour - EoaGroundAge + ATPsetting.EOAvalidityTime)
  
 else:
  
 NewValidity = (ATPtime(k) - EoaGroundAge + ATPsetting.EOAvalidityTime
  
 - (OtherATPmaxTime(k) - NewEoaLoopHour))
  
 if (Message.IsMoreRecent(NewValidity, ATPtime(k))
  
 and (Message.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 shows the survival time of the variants in ZC.

def VariantGroundAge(lineSec, k):
  
 VariantGroundAge = round.ceil((EOA\_Report.MessageContainerCreationTime
  
 - VariantReport(lineSec).CreationTime)
  
 \* SYNCHRODATE\_TIME\_UNIT\_MS / ATP\_CYCLE\_TIME\_MS)
  
 return 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(LineSec, k):
  
 return Message.Received(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 | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | 变量过期时间 | Variants |

def ReceivedVariantReport(LineSec, k):
  
 if (ZCmessageReady(k)
  
 and VariantReportReceived(LineSec, k)):
  
 if (Message.ReplyLocalCc(ReceivedEOAreport(k).CcLoopHour)):
  
 NewValidity = (ReceivedEOAreport(k).CcLoopHour
  
 - VariantGroundAge(LineSec, k)
  
 + ATPsetting.VariantsCBTCvalidityTime)
  
 else:
  
 NewValidity = (ATPtime(k) - VariantGroundAge(LineSec, k)
  
 + ATPsetting.VariantsCBTCvalidityTime
  
 - (OtherATPmaxTime(k) - ReceivedEOAreport(k).CcLoopHour))
  
 if (Message.IsMoreRecent(NewValidity, ATPtime(k))
  
 and (Message.IsMoreRecent
  
 (NewValidity, ReceivedVariantReport[LineSec](k-1).ValidityTime))):
  
 ReceivedVariantReport[LineSec].ValidityTime = NewValidity
  
 ReceivedVariantReport[LineSec].Variants = VariantReport(LineSec, Variants)
  
 else:
  
 ReceivedVariantReport[LineSec] = ReceivedVariantReport[LineSec](k-1)
  
 else:
  
 ReceivedVariantReport[LineSec] = ReceivedVariantReport[LineSec](k-1)
  
 return 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 message. The variants sending from ZC are categorized by line section. Moreover, it will send the line section variants in the current 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(Variant, k):
  
 if (ReceivedVariantReport(Variant.LineSec.Id, k).ValidityTime &gt; ATPtime(k)):
  
 return 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(Variant, k):
  
 if (BlockModeUsed(k)):
  
 return BMvariantValue(Variant, k)
  
 else:
  
 return 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 | √ | × | REF BOOLEAN BOOLEAN |
| VersiionAuthorizedByLC | √ | √ | REF BOOLEAN BOOLEAN |

## F2-Measure Train Kinematics

Figure 56 SART modeling of function F2

## F21-Manage Interface with VPB

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| CBKRead | External | REF \_Ref379961811 \h Interface with VPB |
| CBKWrite | External | REF \_Ref379961811 \h Interface with VPB |
| ImmediateNb | Internal | REF \_Ref345055971 \h F82-Manage Vital Time |
| VitalTime | Internal | REF \_Ref345055971 \h F82-Manage Vital Time |
| WheelFilteredStopped | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| BeaconChecksumFailure | √ | √ | BOOLEAN |
| BeaconMsgByte | √ | × | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| BeaconMsgReady | √ | × | BOOLEAN |
| LockedBeaconMsgReady | √ | √ | BOOLEAN |
| LockedOdometer | √ | √ | REF ST\_ODOMETER\_IMM \h ST\_ODOMETER\_IMM |
| LockedTopLocCounter | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| OdometerImm | √ | × | REF ST\_ODOMETER\_IMM \h ST\_ODOMETER\_IMM |
| SensorTestFlag | √ | √ | BOOLEAN |
| TopLocCounter | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT 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 56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 56, 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 56 Odometer information in interrupt

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | Logical Type | Description | ST\_ODOMETER\_IMM |
| CogCounter | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | 该中断齿数 | TopLocValid |
| REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | 该中断检测到Top-loc | CogCode | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| 该中断齿号寄存器值 | CalCogCounter | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | 当top-loc发生时VPB锁存的齿数 |
| SensorTesting | REF BOOLEAN \h \* MERGEFORMAT BOOLEAN | 该中断是否进行传感器测试 | TestResult |
| REF ENUM\_SENSOR\_TEST\_RESULT \h \* MERGEFORMAT ENUM\_SENSOR\_TEST\_RESULT | 该中断传感器测试结果 | A1 | REF ENUM\_SENSOR\_STATUS \h \* MERGEFORMAT ENUM\_SENSOR\_STATUS |
| 该中断中传感器1的状态 | A2 | REF ENUM\_SENSOR\_STATUS \h \* MERGEFORMAT ENUM\_SENSOR\_STATUS | 该中断中传感器2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 56所示。
OdometerImm records the changes of VPBWrite register in each interrupt, structured as Table 56.
#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.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，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 detected by VPB board regarded as True, OdometerImm.TopLocValid and BeaconMsgReady set as True. Adding TopLocCounter, the data from beacon savor is saved into 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.CogCounterReg未发生变化；
且之前SensorTestFlag为False；
且当前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 regarded as the symbol whether ATP in CPU 1 needs to check the sensor. In the interval, if the value of CBKWrite.CogCounterReg is not changed in the continuous SENSOR\_TEST\_START\_TIME, and the WheelFilteredStopped was False at last cycle, the ATP of CPU1 in VLE-2 board need to set 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.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 of pervious end of cycle as True, it need to set the value of 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 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，A2和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, A2, 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 &amp; Ci=LOW\_LEVEL，Ai = SENSOR\_BLOCKED
if D(i-1)= POWER\_ON &amp; Ci=HIGH\_LEVEL，Ai = SENSOR\_CONDUCT
if D(i-1)= POWER\_OFF &amp; Ci=LOW\_LEVEL，保持上次测试的状态，Ai = Ai(t-1)
if D(i-1)= POWER\_OFF &amp; 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 &amp; Ci=LOW\_LEVEL，Ai = SENSOR\_BLOCKED
if D(i-1)= POWER\_ON &amp; Ci=HIGH\_LEVEL，Ai = SENSOR\_CONDUCT
if D(i-1)= POWER\_OFF &amp; Ci=LOW\_LEVEL，keeps Ai as last status: Ai = Ai(t-1)
if D(i-1)= POWER\_OFF &amp; 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.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.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.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.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.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.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.TestResultas 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.TestResultas 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.TestResultas 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.TestResultas TEST\_STOPPING;
If the TEST\_STOPPING has lasted more than SENSOR\_TEST\_IMMOBILE\_THRESHOLD, the ATP consider the odometer standstill, and set OdometerImm.TestResultas 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.TestResultas 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.SensorTesting标志为False，并设置本周期的SensorTestFlag为False。
When the odometer test result is either TEST\_INCONSISTENT or TEST\_FLOATING, ATP shall stop the sensor test and set OdometerImm.SensorTesting as False for this interrupt, and set 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到数组LockedOdometer[ATP\_INTERRUPT\_NB ]中，其下标为所在中断的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 of all intervals need to be saved into LockedOdometer[ATP\_INTERRUPT\_NB] with the index as ImmediateNb; The value of BeaconMsgReady, TopLocCounter and BeaconMsgByte should be recorded by using LockedBeaconMsgReady, LockedTopLocCounter and LockedBeaconMsgBytefor 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信息：
ATP应检查top-loc发生时VPB锁存的门闩寄存器锁存值CalCogCounter是否在该中断的CogCounter和上个中断的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的依据，而后一个中断作为计算CogPositionAfterTopLoc的依据；
如果两个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:
If the 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.
For TopLocValid:
Only
the cog counter latched by TOPLOC is between the before and after cog counter , TOPLOC is considered validIf the top-loc happened in adjacent interrupt between two CPUs, the former one shall use to calculate the CogPositionBeforeTopLoc, and the latter to calculate the 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为True，则需对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为True
如果校验正确，则设置 BeaconChecksumFailure为False。
如果本周期LockedBeaconMsgReady为False，则设置BeaconChecksumFailure为False。
BeaconChecksumFailure judges whether the checksum of beacon message is correct or not.
In the main task, if LockedBeaconMsgReady is True, LockedBeaconMsgByteneed to be detected, including validity of 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状态为True，且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 in the interrupt is True, and the BeaconChecksumFailure is False, ATP shall set the 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.
#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 represents the accumulated number of received Top-loc signal from power on to current cycle.

if (Initialization)
  
 BeaconCount = 0
  
 else:
  
 BeaconCount = LockedTopLocCounter(k) + 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为True，则解析锁存的LockedBeaconMsgByte数组，生成BeaconMessage信息，其结构为ST\_BEACON\_MSG：
If the BeaconMessageReceive is True, ATP shall parse the value of LockedBeaconMsgByte and generate BeaconMessage with structure as ST\_BEACON\_MSG:

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | Logical Type | Description | ST\_BEACON\_MSG |
| ID | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | 信标标识 | Variants[MAX\_BM\_VARIANT\_NB] |
| REF BOOLEAN \h \* MERGEFORMAT BOOLEAN | 变量状态 | DefaultMessage | REF BOOLEAN \h \* MERGEFORMAT BOOLEAN |
| 是否默认消息 | BlockModeVariantAvailable | REF BOOLEAN \h \* MERGEFORMAT 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为True，则从LockedBeaconMsgByte中获取信标ID，设置BeaconMessage.ID；其他情况保持不变。
If the status of BeaconMessageReceive is True, the BeaconMessage.ID is obtained by LockedBeaconMsgByte; Otherwise, keep it unchanged.

if (Initialization)
  
 BeaconMessage.ID = 0
  
 elif (BeaconMessageReceive(k))
  
 BeaconMessage.ID = LockedBeaconMsgByte[BEACON\_ID\_BITS]
  
 else:
  
 BeaconMessage.ID = 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为True，则从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 is True, the variants is come from LockedBeaconMsgByteand ATP set as BeaconMessage.Variants[MAX\_BM\_VARIANT\_NB]; if there is no beacon read at the end of cycle, there is no changes.

if (Initialization)
  
 BeaconMessage.Variants = {0,..,0}
  
 elif (BeaconMessageReceive(k))
  
 BeaconMessage.Variants(k)
  
 = {LockedBeaconMsgByte[BM\_VARIANTS\_BIT\_0],
  
 ...,
  
 LockedBeaconMsgByte[BM\_VARIANTS\_BIT\_15]}
  
 else:
  
 BeaconMessage.Variants = 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为True，则从LockedBeaconMsgByte中判断是否默认消息，设置BeaconMessage.DefaultMessage；若本周期未读到新的信标则保持不变。
If the BeaconMessageReceive is True, the default message is judged by LockedBeaconMsgByteand ATP set the BeaconMessage.DefaultMessage; if there is no new beacon read, it keeps unchanged.

if (Initialization)
  
 BeaconMessage.DefaultMessage = False
  
 elif (BeaconMessageReceive(k))
  
 BeaconMessage.DefaultMessage(k)
  
 = LockedBeaconMsgByte[DEFAULT\_MESSAGE\_BIT]
  
 else:
  
 BeaconMessage.DefaultMessage = BeaconMessage.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为True，则从LockedBeaconMsgByte中判断变量是否可用信息，设置BeaconMessage.BlockModeVariantAvailable；若本周期未读到新的信标则保持不变。
If the BeaconMessageReceive is True, it is feasible to judge whether the variants are available through LockedBeaconMsgByte and ATP set as BeaconMessage.BlockModeVariantAvailable; If there is no new beacon read, it keeps invariable.

if (Initialization)
  
 BeaconMessage.BlockModeVariantAvailable = False
  
 elif (BeaconMessageReceive(k))
  
 BeaconMessage.BlockModeVariantAvailable(k)
  
 = LockedBeaconMsgByte[BLOCK\_MODE\_VARIANT\_AVAILABLE\_BIT]
  
 else:
  
 BeaconMessage.BlockModeVariantAvailable(k)
  
 = BeaconMessage.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 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.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 | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| BeaconCount | × | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| BeaconMessage | √ | √ | REF ST\_BEACON\_MSG \h ST\_BEACON\_MSG |
| BeaconMessageReceive | √ | √ | BOOLEAN |
| IdenticalLockedOdometer | √ | √ | BOOLEAN |
| LockedBeaconMsgByte | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |

## F22-Monitor the Odometer

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| CoreId | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| IdenticalLockedOdometer | Internal | REF \_Ref345055658 \h F21-Manage Interface with |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| CompCogCode | √ | × | REF NUMERIC\_32 \h NUMERIC\_32 |
| MaxCountCogsRunInCycleExceeded | √ | √ | REF BOOLEAN BOOLEAN |
| SensorSequenceDetected\_1 | √ | √ | REF BOOLEAN BOOLEAN |
| SensorSequenceDetected\_2 | √ | √ | REF BOOLEAN BOOLEAN |
| SensorSequenceDetected\_3 | √ | √ | REF BOOLEAN 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 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 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 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, &quot;1&quot; meaning conduction and &quot;0&quot; 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中锁存各中断的CogCode和ATP主任务计算的期望齿号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）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 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 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 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中锁存的最后一个中断的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 used as the odometer cog value in one deferred task, which is the difference of the CogCounter in the last interrupt of adjacent cycle. The calculation of the TeethCounter shall consider the installation direction of the odometer and the register range of the CogCounter.

TeethCounter(k)
  
 = TeethCounter(k-1)
  
 + (IdenticalLockedOdometer[ATP\_INTERRUPT\_NB - 1].CogCounter(k)
  
 - IdenticalLockedOdometer[ATP\_INTERRUPT\_NB - 1].CogCounter(k-1))
  
 \* ATPsetting.CCcoreOdoCogIncreasing[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是有符号值。如果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 is a signed value. If 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来更新CogPositionBeforeTopLoc；
使用Top-loc发生时中断的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, the CogCounter in the interrupt just before the top-loc happen;
CogPositionAfterTopLoc, the CogCounter in the interrupt when the top-loc happen.

CogPositionBeforeTopLoc(k)
  
 = TeethCounter(k-1)
  
 + ((IdenticalLockedOdometer[i-1].CogCounter(k)
  
 - IdenticalLockedOdometer[ATP\_INTERRUPT\_NB-1].CogCounter(k-1))
  
 \* ATPsetting.CCcoreOdoCogIncreasing[CoreId]))
  
 CogPositionAfterTopLoc(k)
  
 = TeethCounter(k-1)
  
 + ((IdenticalLockedOdometer[i].CogCounter(k)
  
 - IdenticalLockedOdometer[ATP\_INTERRUPT\_NB-1].CogCounter(k-1))
  
 \* ATPsetting.CCcoreOdoCogIncreasing[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. Otherwise, set 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\_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为True；否则，设置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) 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 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(k) == True)
  
 WheelStopped(k)
  
 = ((UnconsistentSensorTest(k) == False)
  
 and (SensorSequenceDetected\_1 = SensorSequenceDetected\_1(k-1))
  
 and (SensorSequenceDetected\_2 = SensorSequenceDetected\_2(k-1))
  
 and (SensorSequenceDetected\_3 = SensorSequenceDetected\_3(k-1)))
  
 else:
  
 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 shall change from False to True on raising edge of WheelStopped information, That is, if:
WheelStopped information was False at cycle k-1,
and WheelStopped information was True at cycle k.
and then:
LastStopCogPosition is assigned to TeethCounter,
At cycle k, WheelFilteredStopped shall change from True to False, according following expression:
the cog moved more than one cog;

def WheelFilteredStopped(k):
  
 if (not WheelFilteredStopped(k-1)
  
 and not WheelStopped(k-1)
  
 and WheelStopped(k)):
  
 LastStopCogPosition = TeethCounter(k)
  
 return True
  
 elif (WheelFilteredStopped(k-1)
  
 and not UnconsistentSensorTest(k)
  
 and abs(TeethCounter(k) - LastStopCogPosition) &lt;= 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(k):
  
 if (abs(IdenticalLockedOdometer[ATP\_INTERRUPT\_NB-1].CogCounter(k)
  
 - IdenticalLockedOdometer[ATP\_INTERRUPT\_NB-1].CogCounter(k-1))
  
 &gt; ATPsetting.OdoMaxCogOnCycle):
  
 return True
  
 else:
  
 for i in range(ATP\_INTERRUPT\_NB-1):
  
 if (abs(IdenticalLockedOdometer[i].CogCounter
  
 - IdenticalLockedOdometer[i+1].CogCounter)&gt; 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.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.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 = 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 | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| CogPositionBeforeTopLoc | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| OdometerCogPositionReady | √ | √ | REF BOOLEAN BOOLEAN |
| SensorTestPerformed | √ | √ | REF BOOLEAN BOOLEAN |
| TeethCounter | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| UnconsistentSensorTest | √ | √ | REF BOOLEAN BOOLEAN |
| WheelFilteredStopped | √ | √ | REF BOOLEAN BOOLEAN |
| WheelKinematicsInvalidForCogCount | √ | √ | REF BOOLEAN BOOLEAN |
| WheelStopped | √ | √ | REF BOOLEAN BOOLEAN |

## F23-Manage the Odometer State

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| CoreId | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| OdometerCogPositionReady | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| MaxCogCalibration | Internal | REF \_Ref345055715 \h F28-Calibrate Wheel Movement |
| MinCogCalibration | Internal | REF \_Ref345055715 \h F28-Calibrate Wheel Movement |
| SensorTestPerformed | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| TeethCounter | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| WheelFilteredStopped | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| WheelKinematicsInvalidForCogCount | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| UnconsistentSensorTest | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| InitializationTimer | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| NoCommunicationWithOdometer | √ | √ | REF BOOLEAN BOOLEAN |
| SensorTestContradiction | √ | √ | REF BOOLEAN 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 57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 57 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 57 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 is &quot;NOT\_INITIALIZED &quot;.
#Category=Functional
#Contribution=SIL4
#Allocation=ATP Software
#Source=[iTC\_CC-SyAD-0149]
[End]
[iTC\_CC\_ATP-SwRS-0176]
If OdometerState is NOT\_INITIALIZED at cycle k, and if wheel detected stopped at cycle k, then WheelMinimumMovement and WheelMaximumMovement shall be set to zero.

if (OdometerState(k) == NOT\_INITIALIZED)
  
 if (WheelFilteredStopped(k) == True))
  
 WheelMinimumMovement = 0
  
 WheelMaximumMovement = 0
  
 else:
  
 WheelMinimumMovement = â€”ATPsetting.MaxMotionPerCycle
  
 WheelMaximumMovement = 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；
本周期未WheelFilteredStopped而且未检测到传感器测试失败
At cycle k, ATP shall consider that OdometerState changes from NOT\_INITIALIZED to WAITING\_COG\_POSITION\_CODE\_READY if:
a falling edge is detected on WheelFilteredStopped information,
and sensors test is consistent at cycle k and was consistent at cycle k-1,

if ((OdometerState(k-1) = NOT\_INITIALIZED)
  
 and (not WheelFilteredStopped(k) and not UnconsistentSensorTest(k))
  
 and (WheelFilteredStopped(k-1))
  
 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 changes from NOT\_INITIALIZED to INVALID
if sensors test is not consistent at cycle k.

if ((OdometerState(k-1) = NOT\_INITIALIZED)
  
 and UnconsistentSensorTest(k))
  
 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(k-1) == WAITING\_COG\_POSITION\_CODE\_READY
  
 and OdometerState(k) == WAITING\_COG\_POSITION\_CODE\_READY)
  
 InitializationTimer = InitializationTimer(k-1) + 1
  
 elif (OdometerState(k-1) != WAITING\_COG\_POSITION\_CODE\_READY
  
 and OdometerState(k) == WAITING\_COG\_POSITION\_CODE\_READY)
  
 InitializationTimer = 1
  
 else:
  
 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 changes from WAITING\_COG\_POSITION\_CODE\_READY to NOT\_INITIALIZED if:
wheel is detected stopped (WheelFilteredStopped),
and cog position remains unknown (not OdometerCogPositionReady),
and there is no sensors test inconsistency,
and time elapsed since last time OdometerState was NOT\_INITIALIZED (InitializationTimer) is strictly less than ATPsetting.OdoInitTimeout

if (OdometerState(k-1) = WAITING\_COG\_POSITION\_CODE\_READY)
  
 and WheelFilteredStopped(k)
  
 and not OdometerCogPositionReady(k)
  
 and not UnconsistentSensorTest(k)
  
 and (InitializationTimer(k) &lt; ATPsetting.OdoInitTimeout)
  
 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 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 was NOT\_INITIALIZED (InitializationTimer) is strictly less than ATPsetting.OdoInitTimeout.

if (OdometerState(k-1) == WAITING\_COG\_POSITION\_CODE\_READY
  
 and OdometerCogPositionReady(k)
  
 and not UnconsistentSensorTest(k)
  
 and (InitializationTimer(k)&lt; ATPsetting.OdoInitTimeout))
  
 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 changes from WAITING\_COG\_POSITION\_CODE\_READY to INVALID if:
sensors test inconsistency is detected,
or time elapsed since last time OdometerState was NOT\_INITIALIZED (InitializationTimer) is more than or equal to the ATPsetting.OdoInitTimeout

if (OdometerState(k-1) == WAITING\_COG\_POSITION\_CODE\_READY)
  
 and ((InitializationTimer(k)&gt;= ATPsetting.OdoInitTimeout)
  
 or UnconsistentSensorTest(k))
  
 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(k) == WAITING\_COG\_POSITION\_CODE\_READY)
  
 if ((TrainFrontEnd(k-1) == END\_2) or (NoUndetectableDanger\_2(k-1) == True))
  
 WheelMinimumMovement(k) = WheelMinimumMovement(k-1) + ATPsetting.MaxMotionPerCycle
  
 WheelMaximumMovement(k) = WheelMaximumMovement(k-1) - ATPsetting.MaxMotionPerCycle
  
 else:
  
 WheelMinimumMovement(k) = WheelMinimumMovement(k-1) â€” ATPsetting.MaxMotionPerCycle
  
 WheelMaximumMovement(k) = WheelMaximumMovement(k-1) + 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 changes from INITIALIZED to INVALID if:
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(k-1) is INITIALIZED
  
 and ((UnconsistentSensorTest(k) == True)
  
 or (not WheelFilteredStopped(k)
  
 and not OdometerCogPositionReady(k)))):
  
 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 and 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(k) is INITIALIZED):
  
 WheelMinimumMovement(k) = MinCogCalibration(k-1) \* (TeethCounter(k) â€” TeethCounter(k-1))
  
 WheelMaximumMovement(k) = MaxCogCalibration(k-1) \* (TeethCounter(k) â€” 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 changes from INVALID to NOT\_INITIALIZED if:
OdometerState was evaluated Invalid at cycle k-1,
and wheel is detected stopped (WheelFilteredStopped),
and there is no sensors test inconsistency.

if (OdometerState(k-1) == INVALID
  
 and WheelFilteredStopped(k)
  
 and not UnconsistentSensorTest(k))
  
 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(k) == INVALID)
  
 WheelMinimumMovement(k) = MinCogCalibration(k-1) \* (TeethCounter(k) â€” TeethCounter(k-1))
  
 WheelMaximumMovement(k) = MaxCogCalibration(k-1) \* (TeethCounter(k) â€” 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.OdoTestContradictionDuration.

def SensorTestContradiction(k):
  
 return (not WheelFilteredStopped(k)
  
 and TeethCounter(k-1) == TeethCounter(k-2)
  
 and not SensorTestPerformed(k))

def NoCommunicationWithOdometer(k):
  
 if (Initialization
  
 or not SensorTestContradiction(k)):
  
 SensorTestContradictionDuration = 0
  
 return False
  
 else:
  
 SensorTestContradictionDuration = SensorTestContradictionDuration(k-1) + 1
  
  
 if (SensorTestContradictionDuration &gt; 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(k)
  
 = ((OdometerState(k) != INVALID)
  
 and (not WheelKinematicsInvalidForCogCount(k))
  
 and (not 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(k):
  
 return round.floor(abs(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(k):
  
 return round.ceil(abs(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 | √ | √ | REF ENUM\_ODOMETER\_STATE \h ENUM\_ODOMETER\_STATE |
| ValidWheelKinematic | √ | √ | REF BOOLEAN BOOLEAN |
| WheelMaxSpeed | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| WheelMaximumMovement | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| WheelMinSpeed | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| WheelMinimumMovement | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |

## F24-Compensate Sliding Slipping Effect

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| MaxCogCalibration | Internal | REF \_Ref345055715 \h F28-Calibrate Wheel Movement |
| OdometerState | Internal | REF \_Ref345055677 \h F23-Manage the Odometer State |
| TeethCounter | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| WheelFilteredStopped | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| WheelMaximumMovement | Internal | REF \_Ref345055677 \h F23-Manage the Odometer State |
| WheelMaxSpeed | Internal | REF \_Ref345055677 \h F23-Manage the Odometer State |
| WheelMinimumMovement | Internal | REF \_Ref345055677 \h F23-Manage the Odometer State |
| WheelMinSpeed | Internal | REF \_Ref345055677 \h F23-Manage the Odometer State |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| AverageWheelAcceleration | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| FilteredWheelAcceleration | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| InstantaneousWheelAcceleration | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| MaxMotionDuringBrakingOrSliding | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| MaxMotionOdometerSignChanged | √ | √ | REF BOOLEAN BOOLEAN |
| MotionUnderEstimationState | √ | √ | REF ENUM\_SLIPPING\_STATE \h ENUM\_SLIPPING\_STATE |
| OdometerAxleMotorized | √ | √ | REF BOOLEAN BOOLEAN |
| OverestimatedMotionMax | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| OverestimatedMotionMin | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| SlidingEnded | √ | √ | REF BOOLEAN BOOLEAN |
| SlidingExcess | √ | √ | REF BOOLEAN BOOLEAN |
| SlipSlideModellingFault | √ | √ | REF BOOLEAN BOOLEAN |
| SlippingEnded | √ | √ | REF BOOLEAN BOOLEAN |
| SlippingExcess | √ | √ | REF BOOLEAN BOOLEAN |
| StartBrakingMovementMax | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| StartBrakingMovementMin | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| StartMotoringMovementMin | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| StartSlidingSpeed | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| StartSlippingSpeed | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| TimeInSliding | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| TimeInSlipping | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| UnderestimatedMotionMax | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| UnderestimatedMotionMin | √ | √ | REF NUMERIC\_32 \h 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(k):
  
 return ((abs(TeethCounter(k) - TeethCounter(k-2))
  
 - abs(TeethCounter(k-2) - TeethCounter(k-4)))
  
 \* MaxCogCalibration(k-1) / 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 measurement is the average of InstantaneousWheelAcceleration over FILTERED\_ACCELERATION\_NB cycles for filtering the fluctuation causing by the sampling period.

FilteredWheelAcceleration(k)
  
 = (InstantaneousWheelAcceleration(k)
  
 + InstantaneousWheelAcceleration(k-1)
  
 + ...
  
 + 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(k)
  
 = (InstantaneousWheelAcceleration(k)
  
 + InstantaneousWheelAcceleration(k-1)
  
 + ...
  
 + 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(k):
  
 if (Initialization
  
 or (MotionOverEstimationState(k) is COASTING)):
  
 return 0
  
 elif ((MotionOverEstimationState(k-1) is COASTING
  
 and MotionOverEstimationState(k) is BRAKING)
  
 or (MotionOverEstimationState(k-1) is COASTING
  
 and MotionOverEstimationState(k) is SLIDING)
  
 or (MotionOverEstimationState(k-1) is BRAKING
  
 and MotionOverEstimationState(k) is SLIDING))):
  
 return MaximumTrainMotion(k-1)
  
 else:
  
 return 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(k-1) &gt; 0)
  
 MaxMotionDuringBrakingOrSliding(k)
  
 = min((MaxMotionDuringBrakingOrSliding(k-1) + WheelMaximumMovement(k)), 0)
  
 elif (StartBrakingMovementMax(k-1) &lt; 0)
  
 MaxMotionDuringBrakingOrSliding(k)
  
 = max((MaxMotionDuringBrakingOrSliding(k-1) + WheelMaximumMovement(k)), 0)
  
 else:
  
 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(k):
  
 return (sign(StartBrakingMovementMax(k-1)) != sign(WheelMaximumMovement(k))
  
 and (abs(MaxMotionDuringBrakingOrSliding(k))
  
 &gt; 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(k) == COASTING)
  
 or (MotionOverEstimationState(k) == BRAKING))
  
 StartSlidingSpeed = 0
  
 elif (((MotionOverEstimationState(k-1) == COASTING)
  
 or (MotionOverEstimationState(k-1) == BRAKING))
  
 and (MotionOverEstimationState(k) == SLIDING))
  
 StartSlidingSpeed = WheelMaxSpeed(k-1)
  
 else:
  
 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(k-1) == SLIDING)
  
 and (MotionOverEstimationState(k) != SLIDING)))
  
 TimeInSliding = 0
  
 elif (MotionOverEstimationState(k) == SLIDING)
  
 TimeInSliding = TimeInSliding(k-1) + 1
  
 else:
  
 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 58所示。
ATP software shall use the over-estimation model for train movement provided by Figure 58 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.BrakingStartAcc (normally -0.3m/s2), ATP shall overestimated the maximum train motion 15% at most.
SLIDING, When the measured acceleration is less than the ATPsetting.SlidingStartAcc (-2.5m/s2 normally) or the average acceleration is less than 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 58 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由COASTING转入BRAKING，并执行：
The state transfers from “COASTING” to “BRAKING” when:

if ((MotionOverEstimationState(k-1) == COASTING)
  
 and (WheelFilteredStopped(k) != True)
  
 and (FilteredWheelAcceleration(k)&lt; ATPsetting.BrakingStartAcc)
  
 and (FilteredWheelAcceleration(k)&gt;= ATPsetting.SlidingStartAcc)
  
 and OdometerState(k) is INITIALIZED)
  
 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 transfers from “COASTING” to “SLIDING” when:

if ((MotionOverEstimationState(k-1) == COASTING)
  
 and (WheelFilteredStopped(k) != True)
  
 and (FilteredWheelAcceleration(k) &lt; ATPsetting.SlidingStartAcc)
  
 and OdometerState(k) is INITIALIZED)
  
 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 transfers from “BRAKING” to “SLIDING” when:

if (MotionOverEstimationState(k-1) == BRAKING
  
 and (FilteredWheelAcceleration(k) &lt; ATPsetting.SlidingStartAcc)
  
 and (AverageWheelAcceleration(k) &lt; ATPsetting.BrakingStartAcc)
  
 and (OdometerState(k) is INITIALIZED)
  
 and (not MaxMotionOdometerSignChanged(k)))
  
 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 transfers from “BRAKING” to “COASTING” when:

if (MotionOverEstimationState(k-1) == BRAKING
  
 and ((AverageWheelAcceleration(k)&gt;= ATPsetting.BrakingStartAcc)
  
 or (OdometerState(k) is INVALID)
  
 or (MaxMotionOdometerSignChanged(k))))
  
 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 transfers from “SLIDING” to “COASTING” when:

if ((MotionOverEstimationState(k-1) == SLIDING)
  
 and ((OdometerState(k) is INVALID)
  
 or (MaxMotionOdometerSignChanged(k) == True))
  
 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.SlidingStartAcc为一个较大的减速度（典型值为-2.3m/s2），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/s2），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/s2) used to detect sliding is far less than the full service braking deceleration (normally -1m/s2), 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) if FilteredWheelAcceleration is strictly less than ATPsetting.SlippingStopAcc and strictly greater than ATPsetting.SlidingStopAcc for more than ATPsetting.SlidingGripRecoveryTime.

if (MotionOverEstimationState(k-1) == SLIDING
  
 and (FilteredWheelAcceleration(k) &lt; ATPsetting.SlippingStopAcc)
  
 and (FilteredWheelAcceleration(k) &gt; ATPsetting.SlidingStopAcc)
  
 and (MotionOverEstimationState(k-2) == SLIDING)
  
 and (FilteredWheelAcceleration(k-1) &lt; ATPsetting.SlippingStopAcc)
  
 and (FilteredWheelAcceleration(k-1) &gt; ATPsetting.SlidingStopAcc)
  
 ...
  
 and (MotionOverEstimationState(k-ATPsetting.SlidingGripRecoveryTime) == SLIDING)
  
 and (FilteredWheelAcceleration(k-ATPsetting.SlidingGripRecoveryTime+1)
  
 &lt; ATPsetting.SlippingStopAcc)
  
 and (FilteredWheelAcceleration(k-ATPsetting.SlidingGripRecoveryTime+1)
  
 &gt; ATPsetting.SlidingStopAcc))
  
 SlidingEnded = True
  
 else:
  
 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 transfers from “SLIDING” to “BRAKING” when:

if ((MotionOverEstimationState(k-1) == SLIDING)
  
 and (OdometerState(k) is INITIALIZED)
  
 and (not MaxMotionOdometerSignChanged(k))
  
 and (TimeInSliding(k-1) &lt;= ATPsetting.SlidingTimeout)
  
 and (0 &lt; (|StartSlidingSpeed(k-1)| + TimeInSliding(k-1)\* ATPsetting.SlidingStopAcc))
  
 and ((|StartSlidingSpeed(k-1)| + TimeInSliding(k-1)\* ATPsetting.SlidingStopAcc)
  
 &lt; |WheelMaxSpeed(k)|)
  
 and (SlidingEnded(k) == True))
  
 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) if FilteredWheelAcceleration is strictly less than ATPsetting.SlippingStopAcc and strictly greater than ATPsetting.SlidingStopAcc for more than ATPsetting.SlidingExcessTime.

if (MotionOverEstimationState(k-1) == SLIDING
  
 and (FilteredWheelAcceleration(k) &lt; ATPsetting.SlippingStopAcc)
  
 and (FilteredWheelAcceleration(k) &gt; ATPsetting.SlidingStopAcc)
  
 and (MotionOverEstimationState(k-2) == SLIDING)
  
 and (FilteredWheelAcceleration(k-1) &lt; ATPsetting.SlippingStopAcc)
  
 and (FilteredWheelAcceleration(k-1) &gt; ATPsetting.SlidingStopAcc)
  
 ...
  
 and (MotionOverEstimationState(k-ATPsetting.SlidingExcessTime) == SLIDING)
  
 and (FilteredWheelAcceleration(k-ATPsetting.SlidingExcessTime+1)
  
 &lt; ATPsetting.SlippingStopAcc)
  
 and (FilteredWheelAcceleration(k-ATPsetting.SlidingExcessTime+1)
  
 &gt; ATPsetting.SlidingStopAcc))
  
 SlidingExcess = True
  
 else:
  
 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 transfers from SLIDING to SKIDDING when:

if (MotionOverEstimationState(k-1) is SLIDING
  
 and OdometerState(k) is INITIALIZED
  
 and not MaxMotionOdometerSignChanged(k)
  
 and (TimeInSliding(k-1) &gt; ATPsetting.SlidingTimeout
  
 or (StartSlidingSpeed(k-1)+ TimeInSliding(k-1) \* ATPsetting.SlidingStopAcc) &lt;= 0
  
 or (((StartSlidingSpeed(k-1) + TimeInSliding(k-1)\* ATPsetting.SlidingStopAcc)
  
 &gt;= WheelMaxSpeed(k))
  
 and SlidingExcess(k))))
  
 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.SlidingStopAcc。因此，如果某周期瞬间加速度小于ATPsetting.SlidingStopAcc；并且在之后的测得加速度满足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.SlidingStopAcc for serious cycles. Therefore, if there was an unexpected instantaneous acceleration less than ATPsetting.SlidingStopAcc, and the after cycles' acceleration met the criteria of SlidingExcess, then the following two conditions can hold simultaneously.

|StartSlidingSpeed(k-1)| + TimeInSliding(k) \* ATPsetting.SlidingStopAcc &gt;= |WheelMaxSpeed(k)|)
  
 AND (SlidingExcess(k) == True)

[iTC\_CC\_ATP-SwRS-0220]
The MotionOverEstimationState transfers from SKIDDING to COASTING when:

if (MotionOverEstimationState(k-1) is SKIDDING
  
 and (WheelFilteredStopped(k)
  
 or OdometerState(k) is INVALID))
  
 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(k):
  
 if (Initialization
  
 or OdometerState(k-1) is not INITIALIZED
  
 or (MotionOverEstimationState(k) is COASTING)):
  
 return 0
  
 elif ((MotionOverEstimationState(k-1) is COASTING
  
 and MotionOverEstimationState(k) is BRAKING)
  
 or (MotionOverEstimationState(k-1) is COASTING
  
 and MotionOverEstimationState(k) is SLIDING)
  
 or (MotionOverEstimationState(k-1) is BRAKING
  
 and MotionOverEstimationState(k) is SLIDING)):
  
 return MinimumTrainMotion(k-1)
  
 else:
  
 return 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(k):
  
 if (sign(StartBrakingMovementMin(k)) == sign(WheelMinimumMovement(k))
  
 and (MotionOverEstimationState (k) is BRAKING
  
 or MotionOverEstimationState (k) is SLIDING)):
  
 if (StartBrakingMovementMin(k) &gt;= 0):
  
 return min(StartBrakingMovementMin(k), WheelMinimumMovement(k))
  
 else:
  
 return (-1 \* min(abs(StartBrakingMovementMin(k)), abs(WheelMinimumMovement(k))))
  
 else:
  
 return 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，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与补偿后的测得位移中绝对值较小的一个，位移方向与StartBrakingMovementMax相同。
In BRAKING state, the maximum train motion overestimated as 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。此时使用测得位移作为最大列车位移；
否则，使用StartBrakingMovementMax
In state SLIDING:
If both WheelMaximumMovement and 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. In this case, ATP shall uses WheelMaximumMovement as current train maximum motion.
Otherwise, uses 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(k):
  
 if MotionOverEstimationState(k) is BRAKING:
  
 if (sign(StartBrakingMovementMax(k)) == sign(WheelMaximumMovement(k))
  
 and abs(WheelMaximumMovement(k)) &gt; abs(StartBrakingMovementMax(k))):
  
 return WheelMaximumMovement(k)
  
 elif (StartBrakingMovementMax(k) &gt;= 0):
  
 return min(abs(StartBrakingMovementMax(k)),
  
 abs(WheelMaximumMovement(k) \* ATPsetting.SlidingCoefficient))
  
 else:
  
 return -1 \* min(abs(StartBrakingMovementMax(k)),
  
 abs(WheelMaximumMovement(k) \* ATPsetting.SlidingCoefficient))
  
 elif MotionOverEstimationState(k) is SLIDING:
  
 if (sign(StartBrakingMovementMax(k)) == sign(WheelMaximumMovement(k))
  
 and abs(WheelMaximumMovement(k)) &gt; abs(StartBrakingMovementMax(k))):
  
 return WheelMaximumMovement(k)
  
 else:
  
 return StartBrakingMovementMax(k)
  
 else: \# Coasting and Skidding
  
 return 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(k):
  
 if (Initialization
  
 or OdometerState(k-1) is not INITIALIZED
  
 or MotionUnderEstimationState(k) is COASTING
  
 or MotionUnderEstimationState(k) is MOTORING):
  
 return 0
  
 elif ((MotionUnderEstimationState(k-1) is COASTING
  
 or MotionUnderEstimationState(k-1) is MOTORING)
  
 and MotionUnderEstimationState(k) is SLIPPING):
  
 return WheelMinSpeed(k-1)
  
 else:
  
 return 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(k):
  
 if (Initialization
  
 or (MotionUnderEstimationState(k-1) is SLIPPING
  
 and MotionUnderEstimationState(k) is not SLIPPING)):
  
 return 0
  
 elif (MotionUnderEstimationState(k) is SLIPPING):
  
 return TimeInSlipping(k-1) + 1
  
 else:
  
 return 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(k):
  
 return not 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 59所示。
ATP software shall use the over-estimation model for train movement provided by Figure 59 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 59 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 transfers from “COASTING” to “MOTORING” when:

if (MotionUnderEstimationState(k-1) is COASTING
  
 and OdometerAxleMotorized(k)
  
 and FilteredWheelAcceleration(k) &lt;= ATPsetting.SlippingStartAcc
  
 and FilteredWheelAcceleration(k) &gt; ATPsetting.TractionStartAcc
  
 and OdometerState(k) is INITIALIZED)
  
 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 transfers from “COASTING” to “SLIPPING” when:

if (MotionUnderEstimationState(k-1) is COASTING
  
 and OdometerAxleMotorized(k)
  
 and FilteredWheelAcceleration(k) &gt; ATPsetting.SlippingStartAcc
  
 and OdometerState(k) is INITIALIZED)
  
 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 transfers from “MOTORING” to “SLIPPING” when:

if (MotionUnderEstimationState(k-1) is MOTORING
  
 and FilteredWheelAcceleration(k) &gt; ATPsetting.SlippingStartAcc
  
 and AverageWheelAcceleration(k) &gt; ATPsetting.MotoringStartAc)
  
 and OdometerState(k) is INITIALIZED
  
 and OdometerAxleMotorized(k))
  
 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 transfers from MOTORING to COASTING” when:

if (MotionUnderEstimationState(k-1) is MOTORING
  
 and (AverageWheelAcceleration(k) &lt;= ATPsetting.TractionStartAcc
  
 or OdometerState(k) is INVALID
  
 or not OdometerAxleMotorized(k)))
  
 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 transfers from SLIPPING to COASTING when:

if (MotionUnderEstimationState(k-1) is SLIPPING
  
 and (OdometerState(k) is INVALID
  
 or not OdometerAxleMotorized(k)))
  
 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(k):
  
 if (MotionUnderEstimationState(k-1) is not SLIPPING):
  
 slipping\_ended\_counter = 0
  
 return False
  
 elif (FilteredWheelAcceleration(k) &lt; ATPsetting.SlippingStopAcc
  
 and FilteredWheelAcceleration(k) &gt; ATPsetting.SlidingStopAcc):
  
 slipping\_ended\_counter(k) = slipping\_ended\_counter(k-1) + 1
  
 return (slipping\_ended\_counter &gt;= 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 transfers from SLIPPING to MOTORING when:

if (MotionUnderEstimationState(k-1) is SLIPPING
  
 and OdometerState(k) is INITIALIZED
  
 and OdometerAxleMotorized(k)
  
 and TimeInSlipping(k-1) &lt;= ATPsetting.SlippingTimeout
  
 and ((StartSlippingSpeed(k-1) + TimeInSlipping(k-1)\* ATPsetting.SlippingStopAcc)
  
 &gt; WheelMinSpeed(k))
  
 and SlippingEnded(k))
  
 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(k):
  
 if (MotionUnderEstimationState(k-1) is not SLIPPING):
  
 slipping\_excess\_counter = 0
  
 return False
  
 elif (FilteredWheelAcceleration(k) &lt; ATPsetting.SlippingStopAcc
  
 and FilteredWheelAcceleration(k) &gt; ATPsetting.SlidingStopAcc):
  
 slipping\_excess\_counter(k) = slipping\_excess\_counter(k-1) + 1
  
 return (slipping\_excess\_counter &gt;= 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 transfers from SLIPPING to SKIDDING when:

if (MotionUnderEstimationState(k-1) is SLIPPING
  
 and OdometerState(k) is INITIALIZED
  
 and OdometerAxleMotorized(k)
  
 and (TimeInSlipping(k-1) &gt; ATPsetting.SlippingTimeout
  
 or (((StartSlippingSpeed(k-1) + TimeInSlipping(k-1)\* ATPsetting.SlippingStopAcc)
  
 &lt;= WheelMinSpeed(k))
  
 and SlippingExcess(k))))
  
 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 transfers from SKIDDING to COASTING when:

if (MotionUnderEstimationState(k-1) is SKIDDING
  
 and (WheelFilteredStopped(k)
  
 or OdometerState(k) is INVALID))
  
 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(k):
  
 if (Initialization
  
 or OdometerState(k-1) is not INITIALIZED
  
 or (MotionUnderEstimationState(k) is COASTING)):
  
 return 0
  
 elif ((MotionUnderEstimationState(k-1) is COASTING
  
 and MotionUnderEstimationState (k) is MOTORING)
  
 or (MotionUnderEstimationState(k-1) is COASTING
  
 and MotionUnderEstimationState (k) is SLIPPING)
  
 or (MotionUnderEstimationState(k-1) is MOTORING
  
 and MotionUnderEstimationState(k) is SLIPPING)):
  
 return MinimumTrainMotion(k-1)
  
 else:
  
 return 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(k):
  
 if MotionUnderEstimationState(k) is MOTORING:
  
 if (WheelMinimumMovement(k) &gt;= 0):
  
 return max(abs(StartMotoringMovementMin(k)),
  
 abs(WheelMinimumMovement(k) \* ATPsetting.SlippingCoefficient))
  
 else:
  
 return -1 \* max(abs(StartMotoringMovementMin(k)),
  
 abs(WheelMinimumMovement(k) \* ATPsetting.SlippingCoefficient))
  
 elif MotionUnderEstimationState(k) is SLIPPING:
  
 return StartMotoringMovementMin(k)
  
 else:
  
 return WheelMinimumMovement(k)

#Category=Functional
#Contribution
=SIL4#Allocation=ATP Software
, Vital Embedded Setting#S
ource=[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(k):
  
 return WheelMaximumMovement(k)

#Category=Functional
#Contribution
=SIL4#Allocation=ATP Software
#S
ource=[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(k)
  
 = ((MotionOverEstimationState(k) != COASTING
  
 and MotionOverEstimationState(k) != BRAKING)
  
 or (MotionUnderEstimationState(k) != COASTING
  
 and 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(k):
  
 if (MotionOverEstimationState(k) is SKIDDING
  
 or MotionUnderEstimationState(k) is SKIDDING
  
  
  
 or ((MotionOverEstimationState(k-1) is BRAKING
  
 or MotionOverEstimationState(k-1) is SLIDING)
  
 and MaxMotionOdometerSignChanged(k)))):
  
 return True
  
 elif (MotionOverEstimationState(k-1) is COASTING
  
 and MotionUnderEstimationState(k-1) is COASTING)
  
 return False
  
 else:
  
 return 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(k):
  
 if (ValidSlipSlideModelling(k-1))
  
 return not SlipSlideModellingFault(k)
  
 elif ((MotionOverEstimationState(k) is COASTING)
  
 and MotionUnderEstimationState(k) is COASTING
  
 and WheelFilteredStopped(k))
  
 return True
  
 else:
  
 return 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(k):
  
 if ValidSlipSlideModelling(k):
  
 return (sign(OverestimatedMotionMax(k))
  
 \* max(abs(OverestimatedMotionMax(k)), abs(UnderestimatedMotionMax(k)))
  
 else:
  
 return 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(k):
  
 if ValidSlipSlideModelling(k):
  
 return (sign(OverestimatedMotionMin(k))
  
 \* min(abs(OverestimatedMotionMin(k)), abs(UnderestimatedMotionMin(k)))
  
 else:
  
 return WheelMinimumMovement(k)

#Category=Functional
#Contribution=SIL4
#Allocation=ATP Software
#Source=[iTC\_CC-SyAD-
0148][End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| MaximumSScompensatedMotion | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| MinimumSScompensatedMotion | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| MotionOverEstimationState | √ | √ | REF ENUM\_SLIDING\_STATE \h ENUM\_SLIDING\_STATE |
| SlipSlideDetected | √ | √ | REF BOOLEAN BOOLEAN |
| ValidSlipSlideModelling | √ | √ | REF BOOLEAN BOOLEAN |

## F25-Calculate Radar Speed

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| CBKWrite | External | REF \_Ref379961811 \h Interface with VPB |
| MaximumSScompensatedMotion | Internal | REF \_Ref345055686 \h F24-Compensate Sliding Slipping Effect |
| MinimumSScompensatedMotion | Internal | REF \_Ref345055686 \h F24-Compensate Sliding Slipping Effect |
| ValidSlipSlideModelling | Internal | REF \_Ref345055686 \h F24-Compensate Sliding Slipping Effect |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| RadarDirection | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| RadarInfo | √ | √ | REF ST\_RADAR\_INFO \h 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 | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | 根据雷达脉冲信息计算的周期位移 | CbkSpeed |
| REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | 根据雷达脉冲信息计算的周期速度 | DrsDistance | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| 根据雷达双通道位移信息累计的运行距离 | DrsMovement | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | 根据雷达双通道位移信息计算的周期位移 |
| DrsSpeed | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | 根据雷达速度信息计算的周期速度 | DrsValid |
| REF BOOLEAN BOOLEAN | 雷达信息有效（TBD，应根据上述信息相互校验） | DrsDirection | REF NUMERIC\_32 \h NUMERIC\_32 |

#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(k):
  
 if RadarInfo(k).DrsValid:
  
 return 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(k):
  
 if not RadarInfo(k).DrsValid:
  
 return 0
  
 elif ((CoreId(k) is END\_1
  
 and RadarInfo(k).DrsDirection &gt; 0)
  
 or (CoreId(k) is END\_2
  
 and RadarInfo(k).DrsDirection &lt;= 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.RadarSpeedThreshold时，最大最小位移将分别按照百分比ATPsetting.RadarDeviationAboveThreshold过估；
当原始测得速度小于配置的阈值ATPsetting.RadarSpeedThreshold时，最大最小位移将按照固定速度值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(k):
  
 if not RadarInfo(k).DrsValid:
  
 return 0
  
 elif RadarRawSpeed(k) &gt;= ATPsetting.RadarSpeedThreshold:
  
 return (ATP\_CYCLE\_TIME \* RadarDirection(k) \* (RadarRawSpeed(k)
  
 + RadarRawSpeed(k) \* ATPsetting.RadarDeviationAboveThreshold / 1000))
  
 else:
  
 return (ATP\_CYCLE\_TIME \* RadarDirection(k)
  
 \* (RadarRawSpeed(k) + 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(k):
  
 if not RadarInfo(k).DrsValid:
  
 return 0
  
 elif RadarRawSpeed(k) &gt;= ATPsetting.RadarSpeedThreshold:
  
 return (ATP\_CYCLE\_TIME \* RadarDirection(k) \* (RadarRawSpeed(k)
  
 - RadarRawSpeed(k) \* ATPsetting.RadarDeviationAboveThreshold / 1000))
  
 else:
  
 return (ATP\_CYCLE\_TIME \* RadarDirection(k)
  
 \* max(0, RadarRawSpeed(k) - 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(k):
  
 return (ATPsetting.RadarApplied
  
 and RadarInfo(k).DrsValid
  
 and (not ValidSlipSlideModelling(k)
  
 or (RadarDirection(k) \* MaximumSScompensatedMotion(k) &gt;= 0
  
 and abs(RadarMotionMin(k)) &lt;= abs(MaximumSScompensatedMotion(k))
  
 and abs(RadarMotionMax(k)) &gt;= abs(MinimumSScompensatedMotion(k)))))

#Category=Functional
#Contribution=SIL4
#Allocation=ATP Software, Vital Embedded Setting
#Source=
[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| RadarMotionMax | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| RadarMotionMin | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| RadarRawSpeed | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| RadarSpeedValid | √ | √ | REF BOOLEAN BOOLEAN |

## F26-Detect Odometer Axle Lock

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATOcontrolTimeValid | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| NonVitalRequest | External | REF \_Ref348357813 \h Interface with CC Non Vital |
| OdometerState | Internal | REF \_Ref345055677 \h F23-Manage the Odometer State |
| RadarRawSpeed | Internal | REF \_Ref379959783 \h F25-Calculate Radar Speed |
| RadarSpeedValid | Internal | REF \_Ref379959783 \h F25-Calculate Radar Speed |
| ValidWheelKinematic | Internal | REF \_Ref345055677 \h F23-Manage the Odometer State |
| WheelMinSpeed | Internal | REF \_Ref345055677 \h F23-Manage the Odometer State |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| AxleLockedDetectionAvailable | √ | √ | REF BOOLEAN BOOLEAN |
| AxlePossiblyLocked | √ | √ | REF BOOLEAN BOOLEAN |
| OdometerRef\_1 | √ | √ | REF ST\_ODOMETER\_REF \h ST\_ODOMETER\_REF |
| OdometerRef\_2 | √ | √ | REF ST\_ODOMETER\_REF \h ST\_ODOMETER\_REF |
| OdometerSpeedUnderThreshold | √ | √ | REF BOOLEAN BOOLEAN |
| ReferenceSpeedUnderThreshold\_1 | √ | √ | REF BOOLEAN BOOLEAN |
| ReferenceSpeedUnderThreshold\_2 | √ | √ | REF BOOLEAN 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 and 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 | REF BOOLEAN \h \* MERGEFORMAT BOOLEAN | 该参考速度可能失效 | PossiblyEnabled |
| REF BOOLEAN \h \* MERGEFORMAT BOOLEAN | 该参考速度可能有效 | OutOfOrder | REF BOOLEAN \h \* MERGEFORMAT BOOLEAN |
| 该参考速度不可用 | Contradictory | REF BOOLEAN \h \* MERGEFORMAT 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(k):
  
 return (ValidWheelKinematic(k)
  
 and 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(k):
  
 return (WheelMinSpeed(k) &lt; 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 defines whether the referenced speed 1 from CCNV is lower than a configurable threshold.

def ReferenceSpeedUnderThreshold\_1(k):
  
 if RadarSpeedValid(k):
  
 return (RadarRawSpeed(k) &lt; ATPsetting.OdoLockedAxleThresholdSpeed)
  
 else:
  
 return (ATOcontrolTimeValid(k)
  
 and 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 defines whether the referenced speed 1 from CCNV is valid or not.

def ReferenceSpeedAvailable\_1(k):
  
 return (RadarSpeedValid(k)
  
 or (ATOcontrolTimeValid(k)
  
 and 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 defines whether the referenced speed 2 from CCNV is lower than a configurable threshold.

if (ATOcontrolTimeValid(k) == True)
  
 ReferenceSpeedUnderThreshold\_2(k) = NonVitalRequest.OdometerRef2SpeedUnderThreshold(k)
  
 else:
  
 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 shows whether the referenced speed 2 from CCNV is effective or not.

if (ATOcontrolTimeValid(k) == True)
  
 ReferenceSpeedAvailable\_2(k) = NonVitalRequest.OdometerRef2Available(k)
  
 else:
  
 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.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),
and 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. PossiblyDisabled(k)
  
 = (ReferenceSpeedAvailable\_1(k)
  
 and ReferenceSpeedUnderThreshold\_1(k)
  
 and OdometerSpeedAvailable(k)
  
 and not 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. 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),
and 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. PossiblyDisabled(k)
  
 = (ReferenceSpeedAvailable\_2(k)
  
 and ReferenceSpeedUnderThreshold\_2(k)
  
 and OdometerSpeedAvailable(k)
  
 and not 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.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 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. PossiblyEnabled(k)
  
 = (ReferenceSpeedAvailable\_1(k)
  
 and not ReferenceSpeedUnderThreshold\_1(k)
  
 and OdometerSpeedAvailable(k)
  
 and not 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.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 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. PossiblyEnabled(k)
  
 = (ReferenceSpeedAvailable\_2(k)
  
 and not ReferenceSpeedUnderThreshold\_2(k)
  
 and OdometerSpeedAvailable(k)
  
 and not 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.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.OdoLockedAxleDisablingLatency.

if (OdometerRef\_1.PossiblyDisabled(k) == True
  
 and OdometerRef\_1.PossiblyDisabled(k-1) == True
  
 ...
  
 and OdometerRef\_1.PossiblyDisabled(k+1-ATPsetting.OdoLockedAxleDisablingLatency) == True)
  
 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.OdoLockedAxleEnablingLatency:

if (OdometerRef\_1.PossiblyEnabled(k) == True
  
 and OdometerRef\_1.PossiblyEnabled(k-1) == True
  
 ...
  
 and OdometerRef\_1.PossiblyEnabled(k+1-ATPsetting.OdoLockedAxleEnablingLatency) == True)
  
 OdometerRef\_1.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.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.OdoLockedAxleDisablingLatency.

if (OdometerRef\_2.PossiblyDisabled(k) == True
  
 and OdometerRef\_2.PossiblyDisabled(k-1) == True
  
 ...
  
 and OdometerRef\_2.PossiblyDisabled(k+1-ATPsetting.OdoLockedAxleDisablingLatency) == True)
  
 OdometerRef\_2.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.OdoLockedAxleEnablingLatency:

if (OdometerRef\_2.PossiblyEnabled(k) == True
  
 and OdometerRef\_2.PossiblyEnabled(k-1) == True
  
 ...
  
 and OdometerRef\_2.PossiblyEnabled(k+1-ATPsetting.OdoLockedAxleEnablingLatency) == True)
  
 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.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)
and 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. Contradictory(k)
  
 = (not OdometerRef\_1.OutOfOrder(k)
  
 and ReferenceSpeedAvailable\_1(k)
  
 and not ReferenceSpeedUnderThreshold\_1(k)
  
 and ValidWheelKinematic(k)
  
 and 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. 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)
and 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.Contradictory (k)
  
 = (not OdometerRef\_2.OutOfOrder(k)
  
 and ReferenceSpeedAvailable\_2(k)
  
 and not ReferenceSpeedUnderThreshold\_2(k)
  
 and ValidWheelKinematic(k)
  
 and 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(k)
  
 = ((OdometerRef\_1.Contradictory(k) and OdometerRef\_2.Contradictory(k))
  
 or (OdometerRef\_1.Contradictory(k)
  
 and (OdometerRef\_2.OutOfOrder(k) or not ReferenceSpeedAvailable\_2(k)))
  
 or (OdometerRef\_2.Contradictory(k)
  
 and (OdometerRef\_1.OutOfOrder(k) or not 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 situation lasts more than ATPsetting.OdoLockedAxleTimeout, the odometer axle shall be considered locked. Once UnrecoverableAxleLocked set to True, it will stay at state True unless ATP re-initialized.

UnrecoverableAxleLocked(k)
  
 = UnrecoverableAxleLocked(k-1)
  
 or (AxlePossiblyLocked(k)
  
 and AxlePossiblyLocked(k-1)
  
 and ...
  
 and AxlePossiblyLocked(k+1-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
  
 = ((not OdometerRef\_1.OutOfOrder and ReferenceSpeedAvailable\_1(k))
  
 or (not OdometerRef\_2.OutOfOrder and 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(k)
  
 = AxleLockedDetectionAvailable(k) and not 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 | √ | √ | REF BOOLEAN BOOLEAN |
| UnrecoverableAxleLocked | √ | √ | REF BOOLEAN BOOLEAN |
| WheelTrainKinematicCorrelation | √ | √ | REF BOOLEAN BOOLEAN |

## F27-Compute Train Kinematics

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| BeaconMessageReceive | Internal | REF \_Ref345055658 \h F21-Manage Interface with |
| CogPositionAfterToploc | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| CogPositionBeforeTopLoc | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| LockedBeaconMsgByte | Internal | REF \_Ref345055658 \h F21-Manage Interface with |
| MaxCogCalibration | Internal | REF \_Ref345055715 \h F28-Calibrate Wheel Movement |
| MaximumSScompensatedMotion | Internal | REF \_Ref345055686 \h F24-Compensate Sliding Slipping Effect |
| MinCogCalibration | Internal | REF \_Ref345055715 \h F28-Calibrate Wheel Movement |
| MinimumSScompensatedMotion | Internal | REF \_Ref345055686 \h F24-Compensate Sliding Slipping Effect |
| MotionOverEstimationState | Internal | REF \_Ref345055686 \h F24-Compensate Sliding Slipping Effect |
| OdometerSpeedAvailable | Internal | REF \_Ref380051973 \h F26-Detect Odometer Axle Lock |
| RadarMotionMax | Internal | REF \_Ref379959783 \h F25-Calculate Radar Speed |
| RadarMotionMin | Internal | REF \_Ref379959783 \h F25-Calculate Radar Speed |
| RadarSpeedValid | Internal | REF \_Ref379959783 \h F25-Calculate Radar Speed |
| TeethCounter | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ValidSlipSlideModelling | Internal | REF \_Ref345055686 \h F24-Compensate Sliding Slipping Effect |
| ValidWheelKinematic | Internal | REF \_Ref345055677 \h F23-Manage the Odometer State |
| WheelFilteredStopped | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| WheelStopped | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| WheelTrainKinematicCorrelation | Internal | REF \_Ref380051973 \h F26-Detect Odometer Axle Lock |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| TrainMinSpeed | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| TrainStopped | √ | √ | REF BOOLEAN 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(k):
  
 return ((ValidSlipSlideModelling(k)
  
 or RadarSpeedValid(k))
  
 and WheelTrainKinematicCorrelation(k)
  
 and 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(k):
  
 if (not RadarSpeedValid(k)):
  
 return MaximumSScompensatedMotion(k)
  
 elif (not ValidSlipSlideModelling(k)):
  
 return RadarMotionMax(k)
  
 else:
  
 return (min(abs(MaximumSScompensatedMotion(k)), abs(RadarSpeedMax(k)))
  
 \* sign(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(k):
  
 if (not RadarSpeedValid(k)):
  
 return MinimumSScompensatedMotion(k)
  
 elif (not ValidSlipSlideModelling(k)):
  
 return RadarMotionMin(k)
  
 else:
  
 return (max(abs(MinimumSScompensatedMotion(k)), abs(RadarSpeedMin(k)))
  
 \* sign(MinimumSScompensatedMotion(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(k)
  
 = ((WheelStopped(k) == True)
  
 and ((MotionOverEstimationState == COASTING)
  
 or (MotionOverEstimationState == BRAKING))
  
 and (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(k)
  
 = ((WheelFilteredStopped(k) == True)
  
 and ((MotionOverEstimationState == COASTING)
  
 or (MotionOverEstimationState == BRAKING))
  
 and (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(k):
  
 if (Initialization):
  
 return False
  
 elif (not TrainHasMoved(k-1)
  
 and not TrainFilteredStopped(k)
  
 and (TeethCounter(k) != TeethCounter(k-1))):
  
 return True
  
 else:
  
 return 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(k)):
  
 return max(0, (round.floor(abs(MinimumTrainMotion(k) / ATP\_CYCLE\_TIME))
  
 + ((ATPsetting.BrakingMinAcc - ATPsetting.MaxGradientAcc)
  
 \* ATP\_CYCLE\_TIME / 2)))
  
 else:
  
 return 0

其中：
ATPsetting.BrakingMinAcc，为列车制动最小减速度，为负值（即绝对值最大）；
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(k):
  
 if (OdometerSpeedAvailable(k)):
  
 return (round.ceil(abs(MaximumTrainMotion(k) / ATP\_CYCLE\_TIME))
  
 + ((ATPsetting.TractionMaxAcc[TrainMinSpeed(k)] + ATPsetting.MaxGradientAcc)
  
 \* ATP\_CYCLE\_TIME / 2))
  
 else:
  
 return round.ceil(abs(MaximumTrainMotion(k)) / ATP\_CYCLE\_TIME)

其中：
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已经大于配置数据ATPsetting.TractionMaxAcc表中的最大速度值，则应当取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(k):
  
 if (BeaconMessageReceive(k)
  
 and ValidTrainKinematic(k)
  
 and not TrainFilteredStopped(k)
  
 and OdometerState(k) is INITIALIZED):
  
 NewBeaconObtained = True
  
 else:
  
 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(k):
  
 if (Initialization):
  
 BeaconBeforeLastObtained = None
  
 elif (NewBeaconObtained(k)):
  
 BeaconBeforeLastObtained = BeaconLastObtained(k-1)
  
 else:
  
 BeaconBeforeLastObtained = BeaconBeforeLastObtained(k-1)
  
 return 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(k):
  
 if (Initialization):
  
 BeaconLastObtained = None
  
 elif (NewBeaconObtained(k)):
  
 BeaconLastObtained = TrackMap.Beacons[LockedBeaconMsgByte.Id]
  
 else:
  
 BeaconLastObtained = BeaconLastObtained(k-1)
  
 return 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(k)):
  
 DistLastBeaconMin(k)= MinCogCalibration(k) \* (TeethCounter(k)- CogPositionAfterTopLoc(k))
  
 DistLastBeaconMax(k)= MaxCogCalibration(k) \* (TeethCounter(k)- CogPositionBeforeTopLoc(k))
  
 else:
  
 DistLastBeaconMin = DistLastBeaconMin(k-1) + MinimumTrainMotion(k)
  
 DistLastBeaconMax = DistLastBeaconMax(k-1) + 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(k):
  
 if (OdometerState(k) is INITIALIZED):
  
 return (MaximumTrainMotion(k) &lt; 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(k):
  
 if (OdometerState(k) is INITIALIZED):
  
 return (MaximumTrainMotion(k) &gt; 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 | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| BeaconLastObtained | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| DistLastBeaconMax | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| DistLastBeaconMin | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| End1RunningForward | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| End2RunningForward | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| MaximumTrainMotion | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| MinimumTrainMotion | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| NewBeaconObtained | √ | √ | REF BOOLEAN BOOLEAN |
| TrainFilteredStopped | √ | √ | REF BOOLEAN BOOLEAN |
| TrainHasMoved | √ | √ | REF BOOLEAN BOOLEAN |
| TrainMaxSpeed | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| ValidTrainKinematic | √ | √ | REF BOOLEAN 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 | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| BeaconMessage | Internal | REF \_Ref345055658 \h F21-Manage Interface with |
| CogPositionAfterTopLoc | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| CogPositionBeforeTopLoc | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |
| End1RunningForward | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| NewBeaconObtained | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| SlipSlideDetected | Internal | REF \_Ref345055686 \h F24-Compensate Sliding Slipping Effect |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| ValidTrainKinematic | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| CalibrationState | √ | √ | REF ENUM\_CALIBRATION\_STATE \h 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 510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 510, 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, 齿距校准状态
The following variables shall update during calibration:
MaxCogCalibration, the overestimated calibration for each cog;
MinCogCalibration, the underestimated calibration for each cog;
CalibrationState, the state of calibration.
Figure 510 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 = CALI\_WAITING
  
 MinCogCalibration = ATPsetting.OdoCaliDefaultCogLengthMin
  
 MaxCogCalibration = 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 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(k-1) = CALI\_WAITING)
  
 and (NewBeaconObtained(k) == True)
  
 and (TrackMap.BeaconBelongsToCalibrationCouple(BeaconMessage.Id(k)) == True)
  
 and (SlipSlideDetected(k) == False))
  
 CalibrationMeasurementStartPositionMin = CogPositionBeforeTopLoc(k)
  
 CalibrationMeasurementStartPositionMax = CogPositionAfterTopLoc(k)
  
 CalibrationMeasurementStartBeacon = BeaconMessage.Id(k)
  
 CalibrationEnd1RunningForward = End1RunningForward(k)
  
 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(k-1) == CALI\_MEASURING)
  
 and (NewBeaconObtained(k) == True)
  
 and (TrackMap.AreNeighbouredBeacons(BeaconMessage.Id(k),
  
 CalibrationMeasurementStartBeacon(k)) == True)
  
 and (SlipSlideDetected(k) == False)
  
 and (CalibrationEnd1RunningForward(k-1) == End1RunningForward(k)))
  
 if ((ATPsetting.OdoCaliDefaultCogLengthMax &gt;= MaxCogCalibrationMeasured)
  
 and (ATPsetting.OdoCaliDefaultCogLengthMin &lt;= MinCogCalibrationMeasured))
  
 CalibrationValidationStartPositionMin = CogPositionBeforeTopLoc(k)
  
 CalibrationValidationStartPositionMax = CogPositionAfterTopLoc(k)
  
 CalibrationValidationStartBeacon = BeaconMessage.Id(k))
  
 CalibrationResultMin = M
  
 inCogCalibrationMeasured CalibrationResultMax = M
  
 axCogCalibrationMeasured CalibrationState = CALI\_VALIDATING
  
 else:
  
 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.MeterCaliMaxMinCalibration[1]
  
 [|CalibrationMeasurementStartPositionMin(k)-CogPositionAfterTopLoc(k)|
  
 - ATPsetting.OdoCaliCogCounterMin]
  
 MaxCogCalibrationMeasured
  
 = CaliMaxRatio \* ATPsetting.MeterCaliMaxMinCalibration[0]
  
 [|CalibrationMeasurementStartPositionMax(k)-CogPositionBeforeTopLoc(k)|
  
 - 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 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(k-1) = CALI\_MEASURING)
  
 and ((ValidTrainKinematic(k) == False)
  
 or (OdometerState(k) is INVALID)
  
 or (SlipSlideDetected(k) == True)
  
 or (CalibrationEnd1RunningForward(k-1) != End1RunningForward(k))
  
 or (TrainFilteredStopped(k) == True)
  
 or ((NewBeaconObtained(k) == True)
  
 and TrackMap.BeaconBelongsToCalibrationCouple(BeaconMessage.Id(k)) == False)))
  
 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 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(k-1) = CALI\_VALIDATING)
  
 and ((ValidTrainKinematic(k) == False)
  
 or (OdometerState(k) is INVALID)
  
 or (SlipSlideDetected(k) == True)
  
 or ((CalibrationEnd1RunningForward(k-1) != End1RunningForward(k))
  
 or (TrainFilteredStopped(k) == True)
  
 or ((NewBeaconObtained(k) == True)
  
 and (TrackMap.IsCalibrationValidationBeacon(BeaconMessage.Id(k),
  
 CalibrationValidationStartBeacon(k)) == False))))
  
 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 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(k-1) = CALI\_VALIDATING)
  
 and (NewBeaconObtained(k) == True)
  
 and (TrackMap.IsCalibrationValidationBeacon(BeaconMessage.Id(k),
  
 CalibrationValidationStartBeacon(k)))
  
 and (SlipSlideDetected(k) == False)
  
 and (CalibrationEnd1RunningForward(k-1) == End1RunningForward(k)))
  
 if ((TrackMap.CalibrationCoupleMaxDistance(BeaconMessage.Id(k),
  
 CalibrationValidationStartBeacon(k)) &lt;= MaxDistanceRanForValidation)
  
 and TrackMap.CalibrationCoupleMinDistance(BeaconMessage.Id(k),
  
 CalibrationValidationStartBeacon(k)) &gt;= MinDistanceRanForValidation)
  
 MinCogCalibration = CalibrationResultMin(k)
  
 MaxCogCalibration = CalibrationResultMax(k)
  
 CalibrationState = CALI\_COMPLETED
  
 else:
  
 CalibrationState = CALI\_WAITING

Where:

MaxDistanceRanForValidation
  
 =(abs(CalibrationValidationStartPositionMin(k)â€” CogPositionAfterTopLoc(k)) + 1)
  
 \* CalibrationResultMax(k)
  
 MinDistanceRanForValidation
  
 =(abs(CalibrationValidationStartPositionMax(k)â€” CogPositionBeforeTopLoc(k)) - 1)
  
 \* 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 | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| MinCogCalibration | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |

## F3-Locate the Train on Track Map

Figure 511 SART modeling of function F3

## F31-Initialize Train Location

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPtime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| BeaconBeforeLastObtained | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| BeaconLastObtained | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| DistLastBeaconMax | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| DistLastBeaconMin | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| End2RunningForward | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MaximumTrainMotion | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MemLocation | External | REF \_Ref348811365 \h Interface with Memorized Location |
| NewBeaconObtained | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| OtherATP | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| SubSystemId | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainHasMoved | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainLocalized | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainLocatedOnKnownPath | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainRealignmentOnBeacon | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainType | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ValidTrainKinematic | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| End2OrientationByBeacon | √ | √ | REF ENUM\_DOT \h ENUM\_DOT |
| MemorizedLocationAuthorized | √ | √ | REF BOOLEAN BOOLEAN |
| MemorizedLocationAvailable | √ | √ | REF BOOLEAN BOOLEAN |
| MemorizedLocationEnable | √ | √ | REF BOOLEAN BOOLEAN |
| TrainStoppedStartTime | √ | √ | REF NUMERIC\_32 \h 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(k):
  
 if (Initialization
  
 or not MovingInitialByBeacon(k)):
  
 return DOT\_UNKNOWN
  
 elif (ATPsetting.PolarizedTrain):
  
 return ATPsetting.End2Orientation
  
 elif (NewBeaconObtained(k)
  
 and TrackMap.AreNeighbouredBeacons(BeaconBeforeLastObtained(k),
  
 BeaconLastObtained(k))):
  
 if (End2RunningForward(k)):
  
 return (TrackMap.OrientationOfNeighbouredBeacons(BeaconBeforeLastObtained(k),
  
 BeaconLastObtained(k)))
  
 else:
  
 return (TrackMap.OrientationOfNeighbouredBeacons(BeaconLastObtained(k),
  
 BeaconBeforeLastObtained(k)))
  
 else:
  
 return 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 512 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 512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 513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 513,which means there are inconsistent orientations for two train ENDs (both END\_1 and END\_2 are toward to UP or DOWN).
Figure 513 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(k):
  
 if (NewBeaconObtained(k)):
  
 if (End2OrientationByBeacon(k) is not None):
  
 BeaconLocation.Ext2 = (TrackMap.LocationUpdateExt2
  
 (End2RunningForward(k),
  
 End2OrientationByBeacon(k),
  
 BeaconLastObtained.Location(k),
  
 MaxMotionOfEnd2(k),
  
 MinMotionOfEnd2(k)))
  
 elif (TrainLocalized(k-1)):
  
 BeaconLocation.Ext2 = (TrackMap.LocationUpdateExt2
  
 (End2RunningForward(k),
  
 TrainLocation(k-1).Ext2.Ort,
  
 BeaconLastObtained.Location(k),
  
 MaxMotionOfEnd2(k),
  
 MinMotionOfEnd2(k)))
  
 else:
  
 BeaconLocation.Ext2 = None
  
  
 BeaconLocation.Uncertainty = (2 \* BeaconLastObtained(k).PositionTolerance
  
 + abs(DistLastBeaconMax(k) - DistLastBeaconMin(k)))
  
 BeaconLocation.Int2 = U
  
 pdateInt2FromExt2 BeaconLocation.Ext1 = U
  
 pdateExt1FromExt2 BeaconLocation.Int1 = U
  
 pdateInt1FromExt2 else:
  
 BeaconLocation = None
  
 return BeaconLocation

其中，所用到的内部函数分别为：

def MaxMotionOfEnd2(k)
  
 return (DistLastBeaconMax(k)
  
 - ATPsetting.CCcoreEnd2BeaconAntennaDistance[CoreId(k)]
  
 - BeaconLastObtained(k).PositionTolerance)
  
 def MinMotionOfEnd2(k)
  
 return (DistLastBeaconMin(k)
  
 - ATPsetting.CCcoreEnd2BeaconAntennaDistance[CoreId(k)]
  
 - BeaconLastObtained(k).PositionTolerance)

UpdateInt2FromExt2，根据Figure 514，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 514，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 514，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 514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 514 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(k):
  
 if (Initialization
  
 or TrainLocalized(k-1)
  
 or TrainFilteredStopped(k)
  
 or (End1RunningForward(k) and not End1RunningForward(k-1))
  
 or (End2RunningForward(k) and not End2RunningForward(k-1))
  
 or abs(DistLastBeaconMax(k)) &gt;= ATPsetting.BeaconPairMaxDistance):
  
 return False
  
 elif (NewBeaconObtained(k)):
  
 return True
  
 else:
  
 return 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(k):
  
 return (MovingInitialByBeacon(k)
  
 and NewBeaconObtained(k)
  
 and (ATPsetting.PolarizedTrain
  
 or (MovingInitialByBeacon(k-1)
  
 and TrackMap.AreNeighbouredBeacons(BeaconBeforeLastObtained(k),
  
 BeaconLastObtained(k))))
  
 and 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(k):
  
 return Offline.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(k):
  
 return (not 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(k):
  
 return (TrainFilteredStopped(k)
  
 and ValidTrainKinematic(k)
  
 and MemorizedLocationAuthorized(k)
  
 and MemorizedLocationEnable(k)
  
 and Message.VitalChecksumValid(MemLocation)
  
 and MemLocation.MemLocVersion == MEM\_LOCATION\_VERSION
  
 and MemLocation.TrainType == TrainType(k)
  
 and MemLocation.TrainId == SubSystemId(k)
  
 and MemLocation.SleepAreaId == (TrackMap.ExistZoneLocationIncluded
  
 (SGL\_SLEEPING\_ZONE, MemLocation(k).Ext2).Id)
  
 and MemLocation.SleepAreaVersion == (TrackMap.ExistZoneLocationIncluded
  
 (SGL\_SLEEPING\_ZONE, MemLocation(k).Ext2).Version)
  
 and MemLocation.TrainLength == 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(k):
  
 if (TrainLocatedOnOtherATP(k-1)):
  
 return OtherATP(k).MotionSinceMemLoc
  
 elif (MemorizedLocationAvailable(k)
  
 or not MemLocationNotConfirmed(k-1)):
  
 return 0
  
 else:
  
 return (MotionSinceMemorizedLocation(k-1) + 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(k):
  
 if (MemorizedLocationAvailable(k)
  
 and not TrainPresumablyLocalized(k-1)):
  
 return True
  
 elif (TrainPresumablyLocalized(k-1)
  
 and (not TrainLocalized(k-1)
  
 or TrainRealignmentOnBeacon(k-1)):
  
 return False
  
 else:
  
 return 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(k):
  
 if (Initialization
  
 or (not TrainFilteredStopped(k-1)
  
 and TrainFilteredStopped(k)):
  
 return ATPtime(k)
  
 else:
  
 return 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(k):
  
 if (OtherATP(k).LocatedOnKnownPath
  
 and TrainFilteredStopped(k)
  
 and OtherATP(k).TrainFilteredStopped
  
 and Message.IsMoreRecent(OtherATP(k).LatestTimeOtherCore, TrainStoppedStartTime(k))
  
 and not TrainLocatedOnKnownPath(k-1)
  
 and not TrainPresumablyLocalized(k)):
  
 TrainLocatedOnOtherATP = True
  
 else:
  
 TrainLocatedOnOtherATP = False
  
 return 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(k):
  
 if (TrainLocatedOnOtherATP(k)):
  
 return OtherATP(k).Location
  
 elif (TrainPresumablyLocalized(k)
  
 and not TrainHasMoved(k)):
  
 return MemLocation(k)
  
 elif (TrainLocatedOnBeacon(k)):
  
 return BeaconLocation(k)
  
 elif (TrainLocalized(k-1)):
  
 return 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 | √ | √ | REF ST\_LOCATION\_TRAIN \h ST\_LOCATION\_TRAIN |
| MotionSinceMemorizedLocation | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| MovingInitialByBeacon | √ | √ | REF BOOLEAN BOOLEAN |
| TrainInitialLocation | √ | √ | REF ST\_LOCATION\_TRAIN \h ST\_LOCATION\_TRAIN |
| TrainLocatedOnBeacon | √ | √ | REF BOOLEAN BOOLEAN |
| TrainLocatedOnOtherATP | √ | √ | REF BOOLEAN BOOLEAN |
| TrainPresumablyLocalized | √ | √ | REF BOOLEAN BOOLEAN |

## F32-Update Train Location

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| BeaconLocation | Internal | REF \_Ref345055728 \h F31-Initialize Train Location |
| BlockModeUsed | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| DistLastBeaconMax | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| End2RunningForward | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MaximumTrainMotion | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MinimumTrainMotion | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| NewBeaconObtained | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| OtherATP | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainInitialLocation | Internal | REF \_Ref345055728 \h F31-Initialize Train Location |
| TrainLocatedOnBeacon | Internal | REF \_Ref345055728 \h F31-Initialize Train Location |
| TrainLocatedOnKnownPath | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainLocatedOnOtherATP | Internal | REF \_Ref345055728 \h F31-Initialize Train Location |
| TrainPresumablyLocalized | Internal | REF \_Ref345055728 \h F31-Initialize Train Location |
| ValidTrainKinematic | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| CBTCwithoutKnownPath | √ | √ | REF BOOLEAN BOOLEAN |
| InverseLocation | √ | √ | REF BOOLEAN BOOLEAN |
| LocalizationFault | √ | √ | REF BOOLEAN BOOLEAN |
| LocationAfterReloc | √ | √ | REF ST\_LOCATION\_TRAIN \h ST\_LOCATION\_TRAIN |
| LocationBeforeReloc | √ | √ | REF ST\_LOCATION\_TRAIN \h ST\_LOCATION\_TRAIN |
| LocationUncertaintyExceed | √ | √ | REF BOOLEAN BOOLEAN |
| LocationUntravelable | √ | √ | REF BOOLEAN BOOLEAN |
| MotionSinceLastReloc | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| RealignmentFailed | √ | √ | REF BOOLEAN BOOLEAN |
| TrainUnitIntegrity | √ | √ | REF BOOLEAN 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(k):
  
 if (TrainLocalized(k-1) and ValidTrainKinematic(k)):
  
 if (OdometerState(k) is INITIALIZED):
  
 if (End2RunningForward(k)):
  
 LocationBeforeReloc.Uncertainty = (TrainLocation(k-1).Uncertainty
  
 - (MaximumTrainMotion(k) - MinimumTrainMotion(k)))
  
 else:
  
 LocationBeforeReloc.Uncertainty = (TrainLocation(k-1).Uncertainty
  
 + (MaximumTrainMotion(k) - MinimumTrainMotion(k)))
  
 else:
  
 LocationBeforeReloc.Uncertainty = (TrainLocation(k-1).Uncertainty
  
 + abs(MaximumTrainMotion(k))
  
 + abs(MinimumTrainMotion(k)))
  
 LocationBeforeReloc.Ext2 = (TrackMap.LocationUpdateExt2(End2RunningForward(k),
  
 TrainLocation(k-1).Ext2.Ort,
  
 TrainLocation(k-1).Ext2,
  
 MaximumTrainMotion(k),
  
 MinimumTrainMotion(k)))
  
 LocationBeforeReloc.Int2 = U
  
 pdateInt2FromExt2 LocationBeforeReloc.Ext1 = U
  
 pdateExt1FromExt2 LocationBeforeReloc.Int1 = U
  
 pdateInt1FromExt2 else:
  
 LocationBeforeReloc = None
  
 return 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(k):
  
 return (TrainLocalized(k-1)
  
 and (not (TrackMap.ReachableBetweenTwoLocations
  
 (LocationBeforeReloc(k).Ext2, LocationBeforeReloc(k).Ext1,
  
 (abs(MaximumTrainMotion) + ATPsetting.LocationTrainLength
  
 + LocationBeforeReloc(k).Uncertainty)))
  
 or not (TrackMap.ReachableBetweenTwoLocations
  
 (LocationBeforeReloc(k).Ext1, LocationBeforeReloc(k).Ext2,
  
 (abs(MaximumTrainMotion) + ATPsetting.LocationTrainLength
  
 + 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(k):
  
 return (LocationBeforeReloc(k).Uncertainty &lt; 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(k):
  
 return (LocationBeforeReloc(k).Uncertainty &gt; 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(k):
  
 return (TrainLocalized(k-1)
  
 and NewBeaconObtained(k)
  
 and (TrackMap.IntersectionOfTwoZones(BeaconLocation.Int2, BeaconLocation.Ext2,
  
 LocationBeforeReloc.Int2, 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(k):
  
 return (TrainLocalized(k-1)
  
 and NewBeaconObtained(k)
  
 and (TrackMap.IntersectionOfTwoZones(BeaconLocation.Int2, BeaconLocation.Ext2,
  
 LocationBeforeReloc.Int2, 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(k):
  
 if (TrainRealignmentOnBeacon(k)):
  
 (LocationAfterReloc.Int2,
  
 LocationAfterReloc.Ext2) = (TrackMap.IntersectionOfTwoZones(LocationBeforeReloc(k).Int2,
  
 LocationBeforeReloc(k).Ext2,
  
 BeaconLocation(k).Int2,
  
 BeaconLocation(k).Ext2))
  
 LocationAfterReloc.Uncertainty = (TrackMap.DistanceBtwTwoLocs(LocationAfterReloc.Int2,
  
 LocationAfterReloc.Ext2,
  
 ATPsetting.LocationMaxUncertaintyConfirmed))
  
 LocationAfterReloc.Int1 = U
  
 pdateInt1FromExt2 LocationAfterReloc.Ext1 = U
  
 pdateExt1FromExt2 elif (not TrainLocalized(k-1)):
  
 LocationAfterReloc = None
  
 else:
  
 LocationAfterReloc = LocationAfterReloc(k-1)
  
 return 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(k):
  
 return (LocPermanentFailure(k-1)
  
 or (TrainLocatedOnKnownPath(k-1)
  
 and 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(k):
  
 if (TrainLocatedOnOtherATP(k)):
  
 return OtherATP(k).MotionSinceLastReloc
  
 elif (TrainRealignmentOnBeacon(k)):
  
 return abs(DistLastBeaconMax(k))
  
 elif (not TrainLocalized(k-1)):
  
 return 0
  
 else:
  
 return (MotionSinceLastReloc(k-1) + abs(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(k):
  
 return (TrainLocalized(k-1)
  
 and not LocationPathKnown(k-1)
  
 and not 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 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(k):
  
 return Offline.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(k):
  
 return (not TrainUnitIntegrity(k)
  
 or not ValidTrainKinematic(k)
  
 or LocationUntravelable(k)
  
 or InverseLocation(k)
  
 or LocationUncertaintyExceed(k)
  
 or RealignmentFailed(k)
  
 or LocPermanentFailure(k)
  
 or (MotionSinceLastReloc(k) &gt; ATPsetting.LocationBeaconValidityDistance)
  
 or (MemLocationNotConfirmed(k-1)
  
 and abs(MotionSinceMemorizedLocation(k)) &gt;= ATPsetting.MemLocValidityDistance)
  
 or CBTCwithoutKnownPath(k)
  
 or 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(k):
  
 if (Initialization
  
 or LocalizationFault(k)):
  
 return False
  
 elif (not TrainLocalized(k-1)
  
 and (TrainPresumablyLocalized(k)
  
 or TrainLocatedOnOtherATP(k)
  
 or TrainLocatedOnBeacon(k))):
  
 return True
  
 else:
  
 return 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(k):
  
 if (not TrainLocalized(k)):
  
 return None
  
 elif (not TrainLocalized(k-1)):
  
 return TrainInitialLocation(k)
  
 elif (TrainRealignmentOnBeacon(k)):
  
 return LocationAfterReloc(k)
  
 else:
  
 return 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 | √ | √ | REF BOOLEAN BOOLEAN |
| TrainLocalized | √ | √ | REF BOOLEAN BOOLEAN |
| TrainLocation | √ | √ | REF ST\_LOCATION\_TRAIN \h ST\_LOCATION\_TRAIN |
| TrainRealignmentOnBeacon | √ | √ | REF BOOLEAN BOOLEAN |

## F33-Confirm Train Localization

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATOcontrolTimeValid | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPtime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| BlockModeEOAvalid | Internal | REF \_Ref345055764 \h F41-Determine the EOA |
| BlockModeUsed | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| End2RunningForward | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MovingInitialByBeacon | Internal | REF \_Ref345055728 \h F31-Initialize Train Location |
| NonVitalRequest | External | REF \_Ref348357813 \h Interface with CC Non Vital |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| OtherATP | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| SubSystemId | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainHasMoved | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainLocalized | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainLocatedOnBeacon | Internal | REF \_Ref345055728 \h F31-Initialize Train Location |
| TrainLocatedOnOtherATP | Internal | REF \_Ref345055728 \h F31-Initialize Train Location |
| TrainLocation | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainPresumablyLocalized | Internal | REF \_Ref345055728 \h F31-Initialize Train Location |
| TrainRealignmentOnBeacon | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainType | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| WheelFilteredStopped | Internal | REF \_Ref345055668 \h F22-Monitor the Odometer |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| ClearingMemLocRequest | √ | √ | REF BOOLEAN BOOLEAN |
| DriverInCab\_1 | √ | √ | REF BOOLEAN BOOLEAN |
| DriverInCab\_2 | √ | √ | REF BOOLEAN BOOLEAN |
| DriverInTrain | √ | √ | REF BOOLEAN BOOLEAN |
| LocationPathKnown | √ | √ | REF BOOLEAN BOOLEAN |
| LocationUncertaintyExceedTime | √ | √ | REF BOOLEAN BOOLEAN |
| MemLocWritten | √ | √ | REF BOOLEAN BOOLEAN |
| NonVitalSelectedFrontEnd | × | √ | REF ENUM\_CAB\_ID \h ENUM\_CAB\_ID |
| WritingMemLocRequest | √ | √ | REF BOOLEAN BOOLEAN |

### Processing

Determine the train frontATP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 or DriverInCab\_2.

def DriverInCab\_1(k):
  
 return Offline.GetDriverInCab\_1(k)

def DriverInCab\_2(k):
  
 return Offline.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(k):
  
 return (DriverInCab\_1(k) is not 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 represents the train front choice from CCNV.

if (ATOcontrolTimeValid(k) == True)
  
 NonVitalSelectedFrontEnd = NonVitalRequest.SelectedFrontEnd(k)
  
 else:
  
 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(k):
  
 if (Initialization):
  
 return END\_2
  
 elif (DriverInTrain(k)):
  
 if (DriverInCab\_1(k)):
  
 return END\_1
  
 else:
  
 return END\_2
  
 elif (NonVitalSelectedFrontEnd(k) is END\_1
  
 or NonVitalSelectedFrontEnd(k) is END\_2):
  
 return NonVitalSelectedFrontEnd(k)
  
 elif (WheelFilteredStopped(k)):
  
 return TrainFrontEnd(k-1)
  
 elif (not 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(k):
  
 if (TrainFrontEnd(k) is END\_2):
  
 return TrainLocation.Ext2.Ort(k)
  
 else:
  
 return 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(k):
  
 if (TrainFrontEnd(k) is END\_1):
  
 TrainFrontLocation.Max = TrainLocation.Ext1
  
 TrainFrontLocation.Min = TrainLocation.Int1
  
 else:
  
 TrainFrontLocation.Max = TrainLocation.Ext2
  
 TrainFrontLocation.Min = TrainLocation.Int2
  
 return 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(k):
  
 if (TrainFrontEnd(k) is END\_1):
  
 TrainRearLocation.Max = TrainLocation.Int2
  
 TrainRearLocation.Min = TrainLocation.Ext2
  
 else:
  
 TrainRearLocation.Max = TrainLocation.Int1
  
 TrainRearLocation.Min = TrainLocation.Ext1
  
 return 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 cannot be TRUE when train cross switch. After train crossed switch and relocalized by beacon successfully, LocationPathKnown can be set to TRUE.

def LocationPathKnown(k):
  
 if (Initialization
  
 or (BlockModeUsed(k)
  
 and TrainLocalized(k)
  
 and not BlockModeEOAvalid(k-1)
  
 and not TrainFilteredStopped(k)
  
 and TrackMap.ExistSwitchBtwTwoLocs(TrainRearLocation(k).Min,
  
 TrainFrontLocation(k).Max))):
  
 return False
  
 elif (TrainLocatedOnBeacon(k)
  
 or TrainRealignmentOnBeacon(k)
  
 or TrainLocatedOnOtherATP(k)
  
 or (TrainPresumablyLocalized(k) and not TrainHasMoved(k))):
  
 return True
  
 else:
  
 return 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。

def TrainLocatedOnKnownPath(k):
  
 return (TrainLocalized(k)
  
 and LocationPathKnown(k))

#Category=Functional
#Contribution=SIL4
#Allocation=ATP Software
#Source=[iTC\_CC-SyAD-1196]
[End]
NOTES:
BM下的EOA与LocationPathKnown和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为True；从而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也不会变为无效。
Figure 515 Train localization state
列车的定位状态及其相关需求编号，如Figure 515 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(k):
  
 if (TrainLocatedOnKnownPath(k)
  
 and not MemLocationNotConfirmed(k)):
  
 return LOCALIZED\_CONFIRMED
  
 elif (TrainLocalized(k)):
  
 return LOCALIZED\_NOT\_CONFIRMED
  
 elif (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(k):
  
 if (Initialization
  
 or not TrainLocalized(k)
  
 or TrainRealignmentOnBeacon(k)):
  
 return False
  
 elif (TrainPresumablyLocalized(k)):
  
 return True
  
 elif (TrainLocatedOnOtherATP(k)):
  
 return OtherATP.LocatedWithMemLocation(k)
  
 else:
  
 return 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(k):
  
 if (Initialization):
  
 return 0
  
 elif (TrainLocalized(k)
  
 and ((MemLocationNotConfirmed(k) or not LocationPathKnown(k))
  
 and (TrainLocation(k).Uncertainty
  
 &gt; ATPsetting.LocationMaxUncertaintyNotConfirmed))):
  
 return (ATPtime(k) + ATPsetting.LocReportValidityTime)
  
 else:
  
 return 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(k):
  
 if (Initialization):
  
 return False
  
 elif (TrainLocalized(k)):
  
 return Message.IsMoreRecent(ATPtime(k), LocationUncertaintyExceedTime(k))
  
 else:
  
 return 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.

def LocalizedAuthorizationForSweepping(k):
  
 return (TrainLocalized(k)
  
 and 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(k):
  
 return (LocationPathKnown(k)
  
 and not 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(k):
  
 if (not TrainFilteredStopped(k)):
  
 return None
  
 else:
  
 for SleepZone in (TrackMap.AllSingsBtwTwoLocs(SGL\_SLEEPING\_ZONE,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrainFrontLocation(k).Max)):
  
 if (TrackMap.LocationInZone(TrainFrontLocation(k).Max,
  
 SleepZone.Location,
  
 SleepZone.Length,
  
 SleepZone.Orientation)
  
 and TrackMap.LocationInZone(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(k):
  
 return (TrainHasMoved(k)
  
 and TrainIncludedInSleepingZone(k) is not None
  
 and 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(k):
  
 return (not WritingMemLocRequest(k) and 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(k):
  
 if (WritingMemLocRequest(k)):
  
 MemLocWritten.MemLocVersion = MEM\_LOCATION\_VERSION
  
 MemLocWritten.SleepAreaId = TrainIncludedInSleepingZone(k).Id
  
 MemLocWritten.SleepAreaVersion = TrainIncludedInSleepingZone(k).Version
  
 MemLocWritten.TrainType = TrainType(k)
  
 MemLocWritten.TrainId = SubSystemId(k)
  
 MemLocWritten.Ext2 = TrainLocation.Ext2
  
 MemLocWritten.Ext1 = TrainLocation.Ext1
  
 MemLocWritten.Uncertainty = TrainLocation.Uncertainty
  
 MemLocWritten.TrainLength = ATPsetting.LocationTrainLength
  
 elif (ClearingMemLocRequest(k)):
  
 MemLocWritten.MemLocVersion = None
  
 MemLocWritten.SleepAreaId = None
  
 MemLocWritten.SleepAreaVersion = None
  
 MemLocWritten.TrainType = None
  
 MemLocWritten.TrainId = None
  
 MemLocWritten.Ext2 = None
  
 MemLocWritten.Ext1 = None
  
 MemLocWritten.Uncertainty = None
  
 MemLocWritten.TrainLength = None
  
 else:
  
 MemLocWritten = MemLocWritten(k-1)
  
 return 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 | √ | √ | REF ENUM\_LOCALIZATION\_STATE \h ENUM\_LOCALIZATION\_STATE |
| LocalizedAuthorizationForSweepping | √ | √ | REF BOOLEAN BOOLEAN |
| LocationNotUncertaintyExceed | √ | √ | REF BOOLEAN BOOLEAN |
| MemLocationNotConfirmed | √ | √ | REF BOOLEAN BOOLEAN |
| TrainConfirmedLocalized | √ | √ | REF BOOLEAN BOOLEAN |
| TrainFrontEnd | √ | √ | REF ENUM\_CAB\_ID \h ENUM\_CAB\_ID |
| TrainFrontLocation | √ | √ | REF ST\_LOCATION\_UNIT \h ST\_LOCATION\_UNIT |
| TrainFrontOrientation | √ | √ | REF ENUM\_DOT \h ENUM\_DOT |
| TrainIncludedInSleepingZone | √ | √ | REF BOOLEAN BOOLEAN |
| TrainLocatedOnKnownPath | √ | √ | REF BOOLEAN BOOLEAN |
| TrainRearLocation | √ | √ | REF ST\_LOCATION\_UNIT \h ST\_LOCATION\_UNIT |

## F4-Monitor Train Energy

Figure 516 SART modeling of function F4

## F41-Determine the EOA

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPtime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| BlockModeUsed | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| BMbeaconReadAge | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| BMvariantRemainingTime | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| BMvariantValue | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| BMvariantValidWhileTemporallyValid | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| CBIvariantMoreAvailableThanBeacon | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| CBIvariantReportLastAge | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| CoreId | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| End2RunningForward | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| HazardousMotionOnNonExclusiveRoute | Internal | REF \_Ref345055830 \h F51-Moral Time |
| OtherATPmaxTime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| OtherATPminTime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| ReceivedEOAreport | Internal | REF \_Ref345055594 \h F15-Manage Variants in CBTC Mode |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFrontEnd | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainFrontLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainLocalized | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainLocatedOnKnownPath | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainMaxSpeed | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| BlockModeEOAvalid | √ | √ | REF BOOLEAN BOOLEAN |
| CBIminProductionAgeSinceSSAcrossing | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| ReceivedVariantsAfterEnteredBMinitialZone | √ | √ | REF BOOLEAN BOOLEAN |
| RestrictiveSignalOverrun | √ | √ | REF BOOLEAN BOOLEAN |
| StopAssuredPointCrossed | √ | √ | REF BOOLEAN BOOLEAN |
| TrainEnteredInBMinitialZoneAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| TrainInBMinitialZone | √ | √ | REF BOOLEAN BOOLEAN |
| TrainInSMIzone | √ | √ | REF BOOLEAN 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为True；
本周期使用BM变量；
上周期RestrictiveSignalOverrun为False；
本周期列车位移MaximumTrainMotion向激活的驾驶室方向运行；
本周期列车车头最大定位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, 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 as True.
Train has localized;
And the current type of EOA is BLOCK\_MODE\_EOA;
And 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 as False.

def RestrictiveSignalOverrun(k):
  
 sing = TrackMap.ExistSingBtwTwoLocs(SGL\_SIGNAL, TrainFrontLocation(k-1).Max,
  
 TrainFrontLocation(k).Max)
  
 return (sing is not None
  
 and BMvariantValidWhileTemporallyValid(k)
  
 and ((TrainFrontEnd(k) is END\_2 and End2RunningForward(k))
  
 or (TrainFrontEnd(k) is END\_1 and End1RunningForward(k)))
  
 and not 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 517 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.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(k):
  
 NewBlock = TrackMap.ExistSingularityInZone(SGL\_NEW\_BLOCK, TrainFrontLocation(k).Min,
  
 ATPsetting.BMinitAreaLength)
  
 Signal = (TrackMap.ExistSingularityInReverseZone(SGL\_SIGNAL,
  
 NewBlock.Location,
  
 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(k):
  
 if (TrainInBMinitialZone(k) is None):
  
 TrainEnteredInBMinitialZoneAge = 0
  
 else:
  
 TrainEnteredInBMinitialZoneAge = TrainEnteredInBMinitialZoneAge(k-1) + 1
  
 return 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(Cbi, k):
  
 Bmcp = TrackMap.ExistSingBtwTwoLocs(SGL\_BMCP, TrainFrontLocation(k-1).Max,
  
 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(Cbi, k):
  
 if (Initialization
  
 or CBIminProductionAgeSinceSSAcrossing(k-1)&gt;= REPORT\_AGE\_MAX):
  
 return REPORT\_AGE\_MAX
  
 elif (StopAssuredPointCrossed(Cbi, k)):
  
 return ATPsetting.VariantsBMALSpresenceTimer
  
 else:
  
 return (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(cbi, k):
  
 return min(CBIminProductionAgeSinceSSAcrossing(Cbi, k),
  
 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(k):
  
 CbiId = TrackMap.CbiId(TrainInBMinitialZone(k).Block)
  
 return (TrainInBMinitialZone(k) is not None
  
 and ((CBIvariantMoreAvailableThanBeacon(CbiId, k)
  
 and ((CBIvariantReportLastAge(CbiId, k)
  
 + ATPsetting.VariantsBMproductionLatencyRadio)
  
 &lt; TrainEnteredInBMinitialZoneAge(k)))
  
 or (BMbeaconReadAge(k) + ATPsetting.VariantsBMproductionLatencyBeacon
  
 &lt; 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(k):
  
 if (Initialization
  
 or not BlockModeUsed(k)
  
 or TrainFrontEnd(k) is not TrainFrontEnd(k-1)
  
 or not TrainLocatedOnKnownPath(k)
  
 or HazardousMotionOnNonExclusiveRoute(k)
  
 or RestrictiveSignalOverrun(k)):
  
 return False
  
 elif (not BlockModeEOAvalid(k-1)
  
 and TrainInBMinitialZone(k) is not None
  
 and BMvariantValue(TrainInBMinitialZone.Variant(k), k)
  
 and ReceivedVariantsAfterEnteredBMinitialZone(k)):
  
 return True
  
 else:
  
 return 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(k):
  
 Smi = TrackMap.ExistZoneLocationIncluded(SGL\_SMI\_ZONE, TrainFrontEnd(k).Max)
  
 return (Smi is not None
  
 and TrainMaxSpeed(k) &lt; 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(k):
  
 return (not BlockModeUsed(k)
  
 and ReceivedEOAreport.TrainFrontEnd == TrainFrontEnd(k)
  
 and ((TrainInSMIzone(k)
  
 and (Message.IsMoreRecent(ReceivedEOAreport(k).WithoutSpacing.ValidityTime,
  
 ATPtime(k)))
  
 and (ReceivedEOAreport(k).WithoutSpacing.Location.Block != 0)
  
 and (TrackMap.DistanceBtwTwoLocs(TrainFrontLocation(k).Min,
  
 ReceivedEOAreport(k).WithoutSpacing.Location,
  
 ATPsetting.EOAmaxDistance) is not None))
  
 or (not TrainInSMIzone(k)
  
 and TrainLocatedOnKnownPath(k)
  
 and (Message.IsMoreRecent(ReceivedEOAreport(k).Classic.ValidityTime,
  
 ATPtime(k)))
  
 and (ReceivedEOAreport(k).Classic.Location.BlockId != 0)
  
 and (TrackMap.DistanceBtwTwoLocs(TrainFrontLocation(k).Min,
  
 ReceivedEOAreport(k).Eoa.Location,
  
 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(k):
  
 if (CBTCmodeEOAvalid(k)):
  
 if (TrainInSMIzone(k)):
  
 return ReceivedEOAreport.WithoutSpacing.Location
  
 else:
  
 return 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(k):
  
 if (BlockModeUsed(k)):
  
 return BlockModeEOAvalid(k)
  
 else:
  
 return 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(k):
  
 if (EndOfAuthorityValid(k)
  
 and TrainFrontEnd(k) is END\_1):
  
 TractionAuthorisedSenseEnd1 = True
  
 else:
  
 TractionAuthorisedSenseEnd1 = False
  
 return 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(k):
  
 if (EndOfAuthorityValid(k)
  
 and TrainFrontEnd(k) is END\_2):
  
 TractionAuthorisedSenseEnd2 = True
  
 else:
  
 TractionAuthorisedSenseEnd2 = False
  
 return TractionAuthorisedSenseEnd2

#Category=Functional
#Contribution=SIL4
#Allocation=ATP Software
#Source=[iTC\_CC-SyAD-0772]
[End]

### Outputs

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| BlockModeEOAvalid | √ | √ | REF BOOLEAN \* MERGEFORMAT BOOLEAN |
| CBIminProductionAge | √ | √ | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 |
| CBTCmodeEOAlocation | √ | √ | REF ST\_LOCATION\_UNIT \h \* MERGEFORMAT ST\_LOCATION\_UNIT |
| CBTCmodeEOAvalid | √ | √ | BOOLEAN |
| EndOfAuthorityValid | √ | √ | BOOLEAN |
| TractionAuthorisedSenseEnd1 | √ | √ | BOOLEAN |
| TractionAuthorisedSenseEnd2 | √ | √ | BOOLEAN |

## F42-Manage Coerced Permissive or Restrictive

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| BlockModeEOAvalid | Internal | REF \_Ref345055764 \h F41-Determine the EOA |
| BMvariantRemainingTime | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| BMvariantValidWhileTemporallyValid | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| NonVitalRequest | External | REF \_Ref348357813 \h Interface with CC Non Vital |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainFrontLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| CoercedPermissive\_1 | √ | √ | REF BOOLEAN BOOLEAN |
| CoercedPermissive\_2 | √ | √ | REF BOOLEAN BOOLEAN |
| CoercedPermissive\_3 | √ | √ | REF BOOLEAN BOOLEAN |
| CoercedPermissive\_4 | √ | √ | REF BOOLEAN BOOLEAN |
| CrossedOverlapTimerInitialSignal | √ | √ | REF BOOLEAN BOOLEAN |
| NotCoercedRestrictive\_1 | √ | √ | REF BOOLEAN BOOLEAN |
| NotCoercedRestrictive\_2 | √ | √ | REF BOOLEAN BOOLEAN |
| NotCoercedRestrictive\_3 | √ | √ | REF BOOLEAN BOOLEAN |
| NotCoercedRestrictive\_4 | √ | √ | REF BOOLEAN BOOLEAN |
| OverlapTimer | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| OverlapTimerPermissive | √ | √ | REF BOOLEAN BOOLEAN |

### Processing

Overlap releasable+Overlap状态管理
当选择BM模式，若OverlapTimer大于0，则设置OverlapTimerPermissive为True
若OverlapTimerPermissive为True，则所有信号机的overlap强制建立
Overlap timer管理
BM模式下，车头最大定位通过具有OverlapTimer初始化属性的信号机，则设置OverlapTimer为该信号机的变量有效期，并倒计时；当CC发送OverlapReleasable为允许状态时，并设置OverlapTimer为0；当CC未授权在BM下运行时，也设置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(k):
  
 return Offline.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(k):
  
 return (BMoverlapReleasableSendable(k)
  
 and TrainFilteredStopped(k)
  
 and BlockModeEOAvalid(k)
  
 and 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(k):
  
 Signal = TrackMap.ExistSingBtwTwoLocs(SGL\_SIGNAL, TrainFrontLocation(k-1).Max,
  
 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(k):
  
 if (not BlockModeEOAvalid(k)
  
 or OverlapReleasable(k)):
  
 return 0
  
 elif (BMvariantValidWhileTemporallyValid(k)
  
 and CrossedOverlapTimerInitialSignal(k) is not None):
  
 return BMvariantRemainingTime(CrossedOverlapTimerInitialSignal(k).CBIvariant.Id, k)
  
 else:
  
 return 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(k):
  
 return (OverlapTimer(k) &gt; 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 57所示。
Table 57 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.GetNotCoercedRestrictive\_1(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.GetNotCoercedRestrictive\_2(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.GetNotCoercedRestrictive\_3(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.GetNotCoercedRestrictive\_4(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(ncr, k):
  
 if (ncr is NOT\_COERCED\_RESTRICTIVE\_1):
  
 CoercedRestrictive = not NotCoercedRestrictive\_1(k)
  
 elif (ncr is NOT\_COERCED\_RESTRICTIVE\_2):
  
 CoercedRestrictive = not NotCoercedRestrictive\_2(k)
  
 elif (ncr is NOT\_COERCED\_RESTRICTIVE\_3):
  
 CoercedRestrictive = not NotCoercedRestrictive\_3(k)
  
 elif (ncr is NOT\_COERCED\_RESTRICTIVE\_4):
  
 CoercedRestrictive = not NotCoercedRestrictive\_4(k)
  
 elif (ncr is VARIANTS\_RECEIVED\_FROM\_CBI\_ID):
  
 CoercedRestrictive = not CBIvariantLowValidity(VARIANTS\_RECEIVED\_FROM\_CBI\_ID, k)
  
 else:
  
 CoercedRestrictive = False
  
 return 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 58所示。
Table 58 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 (k):
  
 CoercedPermissive\_1 = Offline.GetCoercedPermissive\_1(k)
  
 return 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(k):
  
 CoercedPermissive\_2 = Offline.GetCoercedPermissive\_2(k)
  
 return 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(k):
  
 CoercedPermissive\_3 = Offline.GetCoercedPermissive\_3(k)
  
 return 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(k):
  
 CoercedPermissive\_4 = Offline.GetCoercedPermissive\_4(k)
  
 return 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(cr, k):
  
 if (cr is COERCED\_PERMISSIVE\_1):
  
 CoercedPermissive = CoercedPermissive\_1(k)
  
 elif (cr is COERCED\_PERMISSIVE\_2):
  
 CoercedPermissive = CoercedPermissive\_2(k)
  
 elif (cr is COERCED\_PERMISSIVE\_3):
  
 CoercedPermissive = CoercedPermissive\_3(k)
  
 elif (cr is COERCED\_PERMISSIVE\_4):
  
 CoercedPermissive = CoercedPermissive\_4(k)
  
 elif (cr is VARIANTS\_OVERLAP\_PERMISSIVE):
  
 CoercedPermissive = OverlapTimerPermissive(k)
  
 else:
  
 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 | √ | √ | REF BOOLEAN BOOLEAN |
| CoercedPermissive | √ | √ | REF BOOLEAN BOOLEAN |
| CoercedRestrictive | √ | √ | REF BOOLEAN BOOLEAN |
| OverlapReleasable | √ | √ | REF BOOLEAN BOOLEAN |

## F43-Manage Temporary Speed Restriction

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPtime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| DateSynchronizationReport | External | REF \_Ref348814411 \h Interface with LC |
| OtherATPmaxTime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| OtherATPminTime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TSRdownloadContent | External | REF \_Ref348814411 \h Interface with LC |
| VersionAuthorization | External | REF \_Ref348814411 \h Interface with LC |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| LastTSRreportAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| TSRreportAvailable | √ | √ | REF BOOLEAN BOOLEAN |
| TSRreportReceived | √ | √ | REF BOOLEAN BOOLEAN |

### Processing

Receiving TSR message[iTC\_CC\_ATP-SwRS-0687]
TSRreportReceived，收到TSR消息

def TSRreportReceived(lcId, k):
  
 return Message.Received(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(lcId, k):
  
 return Message.Available(TSRreportReceived(lcId, k),
  
 TSRdownloadContent.CcLoopHour,
  
 ATPsetting.TSRvalidityTime,
  
 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(lcId, k):
  
 return Message.LastAge(TSRreportAvailable(lcId, k),
  
 TSRdownloadContent.CcLoopHour,
  
 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 TSRNOTES:
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 | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | Expiration time for the TSR message | Blocks[ REF MAX\_BLOCK\_NB \h MAX\_BLOCK\_NB] |

def ReceivedTSRdatabase(lc, k):
  
 if (Initialization
  
 or (Message.Exists(DateSynchronizationReport(lc), k)
  
 and Message.Exists(VersionAuthorization(lc), k)
  
 and not Message.Exists(TSRdownloadContent(lc), k))
  
 or (not Message.IsMoreRecent(ReceivedTSRdatabase(lc, k-1).ValidityTime, ATPtime(k))
  
 and not TSRreportAvailable(k))):
  
 S
  
 etAllBlockAsDefaultTsr elif (TSRreportAvailable(lc, k)):
  
 NewValidity = 0
  
 if (Message.ReplyLocalCC(TSRdownloadContent(lc).CcLoopHour)):
  
 NewValidity = (TSRdownloadContent(lc).CcLoopHour + ATPsetting.TSRvalidityTime)
  
 else:
  
 NewValidity = (ATPtime(k) + ATPsetting.TSRvalidityTime
  
 - (OtherATPmaxTime(k) - TSRdownloadContent(lc).CcLoopHour))
  
 ReceivedTSRdatabase.ValidityTime = NewValidity
  
 for tsr in range(0, TSRdownloadContent(lc).NumberOfTsr):
  
 S
  
 etTsrInFirstBlock S
  
 etTsrInLastBlock for blk in range(TSRdownloadContent(lc).Tsr[tsr].FirstBlockId + 1,
  
 TSRdownloadContent(lc).Tsr[tsr].LastBlockId):
  
 S
  
 etTsrInIntermediateBlock else:
  
 ReceivedTSRdatabase = ReceivedTSRdatabase(lc, k-1)
  
 return 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.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 | √ | × | REF ST\_TSR\_BLOCK \h 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 | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFrontLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainLocalized | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainMaxSpeed | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainRearLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| V1TractionCutoff | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| V2EbApplied | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| X1TractionCutoff | √ | √ | REF NUMERIC\_32 \h 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为True前提下，ATP根据列车当前最大速度TrainMaxSpeed，当前速度下的最大牵引力加速度（使用列车最小速度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.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. TractionMaxAcc；
来自线路地图，是离线工具从当前所在block起始点上游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, is the maximum acceleration of traction at current speed.
, from 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.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 = V2EbApplied \* V2
  
 EbApplied + ATPsetting.MPauthAltitudeMaxErrorEnergy

The 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 | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| X2EbApplied | √ | √ | REF NUMERIC\_32 \h 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 | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| BlockModeUsed | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| CoercedPermissive | Internal | REF \_Ref345055772 \h F42-Manage Coerced Permissive or Restrictive |
| CoercedRestrictive | Internal | REF \_Ref345055772 \h F42-Manage Coerced Permissive or Restrictive |
| EndOfAuthorityValid | Internal | REF \_Ref345055764 \h F41-Determine the EOA |
| CBTCmodeEOAlocation | Internal | REF \_Ref345055764 \h F41-Determine the EOA |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ReceivedTSRdatabase | Internal | REF \_Ref345055780 \h F43-Manage Temporary Speed Restriction |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainEnergy | Internal | REF \_Ref345055788 \h F44-Compute Train Energy |
| TrainFrontOrientation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainRearLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| VariantValue | Internal | REF \_Ref345055594 \h F15-Manage Variants in CBTC Mode |
| VersionAuthorizedByLC | Internal | REF \_Ref345055594 \h F15-Manage Variants in CBTC Mode |
| X2EbApplied | Internal | REF \_Ref345055788 \h F44-Compute Train Energy |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| PointVSLNotExceedBSR | √ | √ | REF BOOLEAN BOOLEAN |
| PointVSLNotExceedCTE | √ | √ | REF BOOLEAN BOOLEAN |
| PointVSLNotExceedEOA | √ | √ | REF BOOLEAN BOOLEAN |
| PointVSLNotExceedOTE | √ | √ | REF BOOLEAN BOOLEAN |
| PointVSLNotExceedOverlap | √ | √ | REF BOOLEAN BOOLEAN |
| PointVSLNotExceedPSD | √ | √ | REF BOOLEAN BOOLEAN |
| PointVSLNotExceedPZ | √ | √ | REF BOOLEAN BOOLEAN |
| PointVSLNotExceedSignal | √ | √ | REF BOOLEAN BOOLEAN |
| PointVSLNotExceedSwitch | √ | √ | REF BOOLEAN BOOLEAN |
| PointVSLNotExceedTSR | √ | √ | REF BOOLEAN BOOLEAN |
| PointVSLNotExceedZC | √ | √ | REF BOOLEAN BOOLEAN |
| TSRcontrolInhibition | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedTrainSpeedLimit | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedBSR | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedCTE | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedEOA | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedOTE | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedOverlap | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedPSD | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedPSR | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedPZ | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedSignal | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedSwitch | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedTSR | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceedZC | √ | √ | REF BOOLEAN 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.MPauthLimitSpeed作为安全速度限制区域。
限制区能量

def ZoneVSLNotExceedTrainSpeedLimit(k):
  
 return (TrainEnergy(k) &lt; pow(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]

PSRPSR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 518 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 518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 518中的PSR2,PSR3和PSR4。

def ZoneVSLnotExceedPSR(k):
  
 for Psr in TrackMap.AllSingsBtwTwoLocs(SGL\_PSR,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 X2EbApplied(k))):
  
 if (TrainEnergy(k) &gt;= pow(Psr.SpeedLimit)
  
 and (not TrackMap.LocationBtwTwoLocs(Psr.Location,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrainRearLocation(k).Min)
  
 or (TrackMap.ExistSingBtwTwoLocs(SGL\_PSR, Psr.Location,
  
 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(SGL\_PSR,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 ATPsetting.EOAmaxDistance)):
  
 if (TrainEnergy(k) &gt;= (pow(Psr.SpeedLimit)
  
 + (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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.EBguaranteedAccNormalGrip或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 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.AllSingsBtwTwoLocs(SGL\_NEW\_BLOCK,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)))):
  
 if (Block.Bsr is not None
  
 and not CoercedPermissive(Block.CoercedPermissive, k)
  
 and not VariantValue(Block.Bsr.Variant, k)
  
 and TrainEnergy(k) &gt;= 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(k):
  
 for Block in (TrackMap.AllSingsInZone(SGL\_NEW\_BLOCK,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 ATPsetting.EOAmaxDistance)):
  
 if (Block.Bsr is not None
  
 and not CoercedPermissive(Block.Bsr.CoercedPermissive, k)
  
 and not VariantValue(Block.Bsr.Variant, k)
  
 and TrainEnergy(k) &gt;= (pow(Block.Bsr.Speed)
  
 + (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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(k):
  
 return Offline.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.AllSingsBtwTwoLocs(SGL\_NEW\_BLOCK,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min)
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)))):
  
 Tsr = TSRonBlock(Blk, TrackMap.OppositeOrientation(TrainFrontOrientation(k)), k)
  
 if (not TSRcontrolInhibition(k)
  
 and Tsr is not None
  
 and not TrackMap.LocationBtwTwoLocs(Tsr.Position,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrainRearLocation(k).Min)
  
 and TrainEnergy(k) &gt;= 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(blockId, direction, k):
  
 if (not ReceivedTSRdatabase.Blocks[blockId].NotRestrictionApplication):
  
 if (direction is UP):
  
 return (ReceivedTSRdatabase.Blocks[blockId].Position[0],
  
 ReceivedTSRdatabase.Blocks[blockId].Value)
  
 else:
  
 return (ReceivedTSRdatabase.Blocks[blockId].Position[1],
  
 ReceivedTSRdatabase.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(k):
  
 for Blk in (TrackMap.AllSingsInZone(SGL\_NEW\_BLOCK,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 ATPsetting.EOAmaxDistance)):
  
 Tsr = TSRonBlock(Blk, TrainFrontOrientation(k), k)
  
 if (not TSRcontrolInhibition(k)
  
 and Tsr is not None
  
 and TrainEnergy(k) &gt;= (pow(Tsr.Value)
  
 + (Energy
  
 .AccumulationPotentialEnergy (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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(SGL\_OPEN\_TRACK\_END, TrainFrontLocation(k).Max,
  
 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.ExistSingularityInZone(SGL\_OPEN\_TRACK\_END,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 ATPsetting.EOAmaxDistance)
  
 if (Ote is not None)
  
 and TrainEnergy(k) &gt;= (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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.ExistSingularityInZone(SGL\_CLOSE\_TRACK\_END, TrainFrontLocation(k).Max,
  
 X2EbApplied(k)))
  
 if (cte is not None
  
 and TrainEnergy(k) &gt;= 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.ExistSingularityInZone(SGL\_CLOSE\_TRACK\_END,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 ATPsetting.EOAmaxDistance))
  
 if (cte is not None
  
 and TrainEnergy(k) &gt;= (pow(cte.SpeedLimit)
  
 + (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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 59 Energy constraint point of signal and overlap end

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal | Overlap | Constraint Point | CoercedRestrictive | VariantValue |
| CoercedPermissive | 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 |

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(SGL\_SIGNAL, TrainFrontLocation(k).Max, X2EbApplied(k)):
  
 if ((CoercedRestrictive(Sig.NotCoercedRestrictive, k)
  
 or not VariantValue(Sig.Variant, k))
  
 and (not CoercedPermissive(Sig.CoercedPermissive, k)
  
 and not 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(SGL\_SIGNAL,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 ATPsetting.EOAmaxDistance)):
  
 if ((CoercedRestrictive(Sig.NotCoercedRestrictive, k)
  
 or not VariantValue(Sig.Variant, k))
  
 and (not CoercedPermissive(Sig.CoercedPermissive, k)
  
 and not VariantValue(Sig.OverlapVariant, k))):
  
 and TrainEnergy(k) &gt;= (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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(SGL\_OVERLAP\_END, TrainFrontLocation(k).Max,
  
 X2EbApplied(k)):
  
 Signal = TrackMap.ExistSingBtwTwoLocs(SGL\_SIGNAL, TrainFrontLocation(k).Max,
  
 Overlap.Location)
  
 if (Signal is not None
  
 and (CoercedRestrictive(Signal.NotCoercedRestrictive, k)
  
 or not VariantValue(Signal.Variant, k))
  
 and (CoercedPermissive(Signal.CoercedPermissive, k)
  
 or 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(SGL\_OVERLAP\_END,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 ATPsetting.EOAmaxDistance)):
  
 Signal = (TrackMap.ExistSingBtwTwoLocs(SGL\_SIGNAL,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 Overlap.Location))
  
 if (Signal is not None
  
 and (CoercedRestrictive(Signal.NotCoercedRestrictive, k)
  
 or not VariantValue(Signal.Variant, k))
  
 and (CoercedPermissive(Signal.CoercedPermissive, k)
  
 or VariantValue(Overlap.Variant, k))
  
 and TrainEnergy(k) &gt;= (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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(TrainFrontLocation(k).Max, X2EbApplied(k)):
  
 if (VariantValue(Switch.Variant1, k) == 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 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 ATPsetting.EOAmaxDistance)):
  
 if (VariantValue(Switch.Variant1, k) == VariantValue(Switch.Variant2, k)
  
 and TrainEnergy(k) &gt;= (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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(SGL\_PROTECTION\_ZONE,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)))):
  
 if ((not TrackMap.LocationBtwTwoLocs(Pz.Location,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrainRearLocation(k).Min)
  
 or TrackMap.LocationInZone(TrainRearLocation(k).Min, Pz.Location, Pz.Length))
  
 and not CoercedPermissive(Pz.CoercedPermissive, k)
  
 and (CoercedRestrictive(Pz.NotCoercedRestrictive, k)
  
 or not 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(SGL\_PROTECTION\_ZONE,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 ATPsetting.EOAmaxDistance)):
  
 if (not CoercedPermissive(Pz.CoercedPermissive, k)
  
 and (CoercedRestrictive(Pz.NotCoercedRestrictive, k)
  
 or not VariantValue(Pz.Variant, k))
  
 and TrainEnergy(k) &gt;= (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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.AllSingsBtwTwoLocs(SGL\_PSD\_ZONE,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)))):
  
 if ((not TrackMap.LocationBtwTwoLocs(Psd.Location,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrainRearLocation(k).Min)
  
 or TrackMap.LocationInZone(TrainRearLocation(k).Min, Psd.Location, Psd.Length))
  
 and not CoercedPermissive(Psd.CoercedPermissive, k)
  
 and (CoercedRestrictive(Psd.NotCoercedRestrictive, k)
  
 or not 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(SGL\_PSD\_ZONE,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k) ),
  
 ATPsetting.EOAmaxDistance)):
  
 if (not CoercedPermissive(Psd.CoercedPermissive, k)
  
 and (CoercedRestrictive(Psd.NotCoercedRestrictive, k)
  
 or not VariantValue(Psd.Variant, k))
  
 and TrainEnergy(k) &gt;= (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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 authorizationCBTC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.AllSingsBtwTwoLocs(SGL\_NEW\_BLOCK, TrainRearLocation(k).Min,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)))):
  
 if (not BlockModeUsed(k)
  
 and not VersionAuthorizedByLC(TrackMap.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(SGL\_NEW\_BLOCK,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k)),
  
 ATPsetting.EOAmaxDistance)):
  
 if (not BlockModeUsed(k)
  
 and not VersionAuthorizedByLC(TrackMap.ZCId(NewBlock.Id), k)
  
 and TrainEnergy(k) &gt;= (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 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]

EOACBTC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(k)
  
 and (TrackMap.LocationBtwTwoLocs(CBTCmodeEOAlocation(k),
  
 TrainFrontLocation(k).Min,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, 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(k)
  
 and not (TrackMap.LocationBtwTwoLocs(CBTCmodeEOAlocation(k),
  
 TrainFrontLocation(k).Min,
  
 TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max, X2EbApplied(k))))
  
 and TrainEnergy(k) &gt;= (Energy.AccumulationPotentialEnergy
  
 (TrackMap.CalculateZoneBorder(TrainFrontEnd(k).Max,
  
 X2EbApplied(k)),
  
 CBTCmodeEOAlocation(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(k):
  
 return (EndOfAuthorityValid(k)
  
 and ZoneVSLNotExceedTrainSpeedLimit(k)
  
 and ZoneVSLnotExceedPSR(k)
  
 and ZoneVSLnotExceedBSR(k)
  
 and ZoneVSLnotExceedTSR(k)
  
 and ZoneVSLnotExceedOTE(k)
  
 and ZoneVSLnotExceedCTE(k)
  
 and ZoneVSLnotExceedSignal(k)
  
 and ZoneVSLnotExceedOverlap(k)
  
 and ZoneVSLnotExceedSwitch(k)
  
 and ZoneVSLnotExceedPZ(k)
  
 and ZoneVSLnotExceedPSD(k)
  
 and ZoneVSLnotExceedZC(k)
  
 and 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(k):
  
 return (EndOfAuthorityValid(k)
  
 and PointVSLnotExceedPSR(k)
  
 and PointVSLnotExceedBSR(k)
  
 and PointVSLnotExceedTSR(k)
  
 and PointVSLnotExceedOTE(k)
  
 and PointVSLnotExceedCTE(k)
  
 and PointVSLnotExceedSignal(k)
  
 and PointVSLnotExceedOverlap(k)
  
 and PointVSLnotExceedSwitch(k)
  
 and PointVSLnotExceedPZ(k)
  
 and PointVSLnotExceedPSD(k)
  
 and PointVSLnotExceedZC(k)
  
 and 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 | √ | √ | REF BOOLEAN BOOLEAN |
| ZoneVSLNotExceed | √ | √ | REF BOOLEAN BOOLEAN |

## F46-Determine Over Energy

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| InhibitEmergencyBrake | Internal | REF \_Ref345055931 \h F71-Outputs to Rolling-stock |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| PointVSLNotExceed | Internal | REF \_Ref345055794 \h F45-Process Singularities |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| ZoneVSLNotExceed | Internal | REF \_Ref345055794 \h F45-Process Singularities |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| TrainEnergyControlDisabled | √ | √ | REF BOOLEAN 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(k):
  
 return Offline.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(k):
  
 return (not ZoneVSLNotExceed(k)
  
 or not 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(k):
  
 return 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(k):
  
 return (TrainPossiblyInOverEnergy(k)
  
 and not TrainEnergyControlDisabled(k)
  
 and (not TrainFilteredStopped(k)
  
 or (TrainFilteredStopped(k)
  
 and (ATPsetting.MPauthImmoBehaviourAtFS
  
 is IB\_APPLY\_EMERGENCY\_BRAKE)
  
 or ((ATPsetting.MPauthImmoBehaviourAtFS
  
 is IB\_APPLY\_EMERGENCY\_BRAKE\_WHEN\_TRIGGERED)
  
 and not 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(k):
  
 return (TrainPossiblyInOverEnergy(k)
  
 and not TrainEnergyControlDisabled
  
 and (TrainFilteredStopped(k)
  
 and (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 | √ | √ | REF BOOLEAN BOOLEAN |
| MotionProtectionInhibition | √ | √ | REF BOOLEAN BOOLEAN |
| PBforOverEnergy | √ | √ | REF BOOLEAN BOOLEAN |
| TrainPossiblyInOverEnergy | √ | √ | REF BOOLEAN BOOLEAN |

## F5-Monitor Train Position and Speed

Figure 519 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 | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| BlockModeUsed | Internal | REF \_Ref345055584 \h F14-Manage Variants in Block Mode |
| InhibitEmergencyBrake | Internal | REF \_Ref345055931 \h F71-Outputs to Rolling-stock |
| MotionProtectionInhibition | Internal | REF \_Ref345055806 \h F46-Determine Over Energy |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainFrontEnd | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainLocatedOnKnownPath | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainLocation | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| VariantValue | Internal | REF \_Ref345055594 \h F15-Manage Variants in CBTC Mode |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| NotOnRestrictiveMoralTimeArea\_1 | √ | √ | REF BOOLEAN BOOLEAN |
| NotOnRestrictiveMoralTimeArea\_2 | √ | √ | REF BOOLEAN 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.MTdistance的一段范围。
当满足下列所有条件时，设置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为False。

def NotOnRestrictiveMoralTimeArea\_1(k):
  
 Signal = TrackMap.ExistSingularityInReverseZone(SGL\_SIGNAL,
  
 TrainLocation.Ext1,
  
 ATPsetting.MTdistance + TrainLocation(k).Uncertainty)
  
 return (TrainLocatedOnKnownPath(k)
  
 and (Signal is None
  
 or not Signal.BmMoralTime
  
 or 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为False。

def NotOnRestrictiveMoralTimeArea\_2(k):
  
 Signal = TrackMap.ExistSingularityInReverseZone(SGL\_SIGNAL,
  
 TrainLocation.Ext2,
  
 ATPsetting.MTdistance + TrainLocation(k).Uncertainty)
  
 return (TrainLocatedOnKnownPath(k)
  
 and (Signal is None
  
 or not Signal.BmMoralTime
  
 or 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, and if this situation lasts more than ATPsetting. MTtimeout cycles, ATP shall consider that route exclusivity is not guaranteed and RouteExclusivityGuaranted\_1 shall be set to False.
If ATP detects that train is NotOnRestrictiveMoralTimeArea\_1, route exclusivity shall consider as guaranteed for that direction of travel and RouteExclusivityGuaranted\_1 shall set to True

def RouteExclusivityGuaranted\_1(k):
  
 if (Initialization):
  
 RouteExclusivityGuaranted\_1 = False
  
 elif (RouteExclusivityGuaranted\_1(k-1)
  
 and not NotOnRestrictiveMoralTimeArea\_1(k)):
  
 if (MoralTimeTimer\_1(k-1) &lt; round.floor(ATPsetting.MTtimeout)):
  
 MoralTimeTimer\_1 = MoralTimeTimer\_1(k-1) + 1
  
 else:
  
 RouteExclusivityGuaranted\_1 = False
  
 elif (NotOnRestrictiveMoralTimeArea\_1(k)):
  
 MoralTimeTimer\_1 = 1
  
 RouteExclusivityGuaranted\_1 = True
  
 else:
  
 RouteExclusivityGuaranted\_1 = RouteExclusivityGuaranted\_1(k-1)
  
 return 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, and if this situation lasts more than ATPsetting.MTtimeout cycles, ATP shall consider that route exclusivity is not guaranteed and RouteExclusivityGuaranted\_2 shall set to False.
If ATP detects that train is NotOnRestrictiveMoralTimeArea\_2, route exclusivity shall consider as guaranteed for that direction of travel and RouteExclusivityGuaranted\_2 shall set to True

def RouteExclusivityGuaranted\_2(k):
  
 if (Initialization):
  
 RouteExclusivityGuaranted\_2 = False
  
 elif (RouteExclusivityGuaranted 2(k-1)
  
 and not NotOnRestrictiveMoralTimeArea\_2(k)):
  
 if (MoralTimeTimer\_2(k-1) &lt; round.floor(ATPsetting.MTtimeout)):
  
 MoralTimeTimer\_2 = MoralTimeTimer\_2(k-1) + 1
  
 else:
  
 RouteExclusivityGuaranted\_2 = False
  
 elif (NotOnRestrictiveMoralTimeArea\_2(k)):
  
 MoralTimeTimer\_2 = 1
  
 RouteExclusivityGuaranted\_2 = True
  
 else:
  
 RouteExclusivityGuaranted\_2 = RouteExclusivityGuaranted\_2(k-1)
  
 return 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 is False, ATP shall request emergency braking if and only if:
TrainFrontEnd is not END\_2,
RM forward nor RM reverse are not selected,
and block mode is not selected.
If RouteExclusivityGuaranted\_2 is False, ATP shall request emergency braking if and only if:
TrainFrontEnd is not END\_1,
RM forward nor RM reverse driving mode are not selected,
and block mode is not selected.

def HazardousMotionOnNonExclusiveRoute(k):
  
 return (((not RouteExclusivityGuaranted\_1(k) and (TrainFrontEnd(k)!= END\_2))
  
 or (not RouteExclusivityGuaranted\_2(k) and (TrainFrontEnd(k)!= END\_1)))
  
 and not MotionProtectionInhibition(k)
  
 and 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(k)
  
 = HazardousMotionOnNonExclusiveRoute(k)
  
 and TrainFilteredStopped(k)
  
 and (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(k)
  
 = ï¼ˆHazardousMotionOnNonExclusiveRoute(k)
  
 and (not TrainFilteredStopped(k)
  
 or (TrainFilteredStopped(k)
  
 and ((ATPsetting.MTimmoBehaviourAtFS == IB\_APPLY\_EMERGENCY\_BRAKE)
  
 or ((ATPsetting.MTimmoBehaviourAtFS == IB\_APPLY\_EMERGENCY\_BRAKE\_WHEN\_TRIGGERED)
  
 and not 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 | √ | √ | REF BOOLEAN BOOLEAN |
| HazardousMotionOnNonExclusiveRoute | √ | √ | REF BOOLEAN BOOLEAN |
| PBonNonExclusiveRoute | √ | √ | REF BOOLEAN BOOLEAN |
| RouteExclusivityGuaranted\_1 | √ | √ | REF BOOLEAN BOOLEAN |
| RouteExclusivityGuaranted\_2 | √ | √ | REF BOOLEAN 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 | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| End2RunningForward | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| InhibitEmergencyBrake | Internal | REF \_Ref345055931 \h F71-Outputs to Rolling-stock |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainFrontLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainLocatedOnKnownPath | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainRearLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| ValidTrainKinematic | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| VariantValue | Internal | REF \_Ref345055594 \h F15-Manage Variants in CBTC Mode |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| NUDEdistanceAccount\_1 | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| NUDEdistanceAccount\_2 | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| NoUndetectableDanger\_1 | √ | √ | REF BOOLEAN BOOLEAN |
| NoUndetectableDanger\_2 | √ | √ | REF BOOLEAN 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 510所示。
Table 510 Logical inputs for vital zone

|  |  |
| --- | --- |
| PzType for PermissiveZoneLogicalInput | NrzType for 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(PzType, k):
  
 return (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.AllSingsBtwTwoLocs(SGL\_VITAL\_ZONE,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrainFrontLocation(k).Max)):
  
 if (Vz.PermissiveZoneLogicalInput is PzType
  
 and TrackMap.LocationInZone(TrainFrontLocation(k).Max,
  
 Vz.Location,
  
 Vz.Length,
  
 Vz.Orientation)
  
 and TrackMap.LocationInZone(TrainRearLocation(k).Min,
  
 Vz.Location,
  
 Vz.Length,
  
 Vz.Orientation)
  
 and (Vz.Variant is None
  
 or (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(NrzType, k):
  
 return (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.AllSingsBtwTwoLocs(SGL\_VITAL\_ZONE,
  
 TrackMap.BlockOrigin(TrainRearLocation(k).Min),
  
 TrainFrontLocation(k).Max)):
  
 if (Vz.NotRestrictiveZoneLogicalInput is NrzType
  
 and (TrackMap.IntersectionOfTwoZones(TrainRearLocation(k).Min,
  
 TrainFrontLocation(k).Max,
  
 vz.Location,
  
 TrackMap.CalculateZoneBorder(vz.Location, vz.Length))
  
 is not None)
  
 and (Vz.Variant is None
  
 or not 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 &quot;undetectable danger risk&quot;, 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(k):
  
 return Offline.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 &quot;No Undetectable Danger in Extremity 2&quot; shall be consider as permissive status according to project configuration.

def NoUndetectableDanger\_2(k):
  
 return Offline.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或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(k):
  
 return (not NoUndetectableDanger\_1(k)
  
 and not 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(k)
  
 = (UndetectableDangerRiskForNoNUDE(k)
  
 and TrainFilteredStopped(k)
  
 and (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 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 MaximumTrainMotionis greater than 0, ATP accumulate the maximum movement in this cycle with the distance of last cycle;
Or if the MaximumTrainMotionis 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(k):
  
 if (INTIALIZATION
  
 or NoUndetectableDanger\_1(k)
  
 or (EBappliedForMotionWithoutNUDE(k-1) and TrainFilteredStopped(k)))
  
 return 0
  
 elif (ValidTrainKinematic(k) != True)
  
 return ATPsetting.NUDEdistanceWithoutMotionAvailable
  
 elif (OdometerState(k) is INITIALIZED)
  
 if (End1RunningForward(k))
  
 return max(0, NUDEdistanceAccount\_1(k-1)+ MaximumTrainMotion(k))
  
 else:
  
 return max(0, NUDEdistanceAccount\_1(k-1)+ MinimumTrainMotion(k))
  
 else:
  
 return 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 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 is less than 0, ATP accumulate the maximum movement in this cycle with the distance of last cycle;
Else: if the 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(k):
  
 if (INTIALIZATION
  
 or NoUndetectableDanger\_2(k)
  
 or (EBappliedForMotionWithoutNUDE(k-1) and TrainFilteredStopped(k)))
  
 return 0
  
 elif (ValidTrainKinematic(k) != True)
  
 return -1 \* ATPsetting.NUDEdistanceWithoutMotionAvailable(k)
  
 elif (OdometerState(k) is INITIALIZED)
  
 if (End2RunningForward(k))
  
 return min(0, NUDEdistanceAccount\_2(k-1)+ MaximumTrainMotion(k))
  
 else:
  
 return min(0, NUDEdistanceAccount\_2(k-1)+ MinimumTrainMotion(k))
  
 else:
  
 return 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(k)
  
 = (not NoUndetectableDanger\_1(k)
  
 and (NUDEdistanceAccount\_1(k)&gt; ATPsetting.NUDElimitDistance))
  
 or (not NoUndetectableDanger\_2(k)
  
 and (NUDEdistanceAccount\_2(k)&lt; -1 \* 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为True时，设置EBappliedForMotionWithoutNUDE为True；
当UndetectDangerMotionWithoutNUDE由True变为False后，还需保持EBappliedForMotionWithoutNUDE 在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.NUDEtrainStopDurationBeforeEBrelease, if the train has moved without NUDE more than project restricted distance.
 When UndetectDangerMotionWithoutNUDE is True, ATP shall set EBappliedForMotionWithoutNUDE to True;
 When UndetectDangerMotionWithoutNUDE change from True to False, ATP shall maintain EBappliedForMotionWithoutNUDE to True in period ATPsetting.NUDEtrainStopDurationBeforeEBrelease；
Over the time, set this value to False.

def EBappliedForMotionWithoutNUDE(k):
  
 if (UndetectDangerMotionWithoutNUDE(k)):
  
 NudeEBreleaseCounter = 0
  
 return True
  
 elif (NudeEBreleaseCounter &lt; ATPsettings.NUDEtrainStopDurationBeforeEBrelease):
  
 NudeEBreleaseCounter(k) = 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 &quot;undetectable danger risk&quot; and has stopped, ATP shall determine whether keeping the EB output according to the project configuration.

EBforUndetectableDangerRisk(k)
  
 = (EBappliedForMotionWithoutNUDE(k)
  
 or (UndetectableDangerRiskForNoNUDE(k)
  
 and (not TrainFilteredStopped(k)
  
 or (ATPsetting.NUDEimmoBehaviourAtFS == IB\_APPLY\_EMERGENCY\_BRAKE)
  
 or ((ATPsetting.NUDEimmoBehaviourAtFS == IB\_APPLY\_EMERGENCY\_BRAKE\_WHEN\_TRIGGERED)
  
 and (not 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 | √ | √ | REF BOOLEAN BOOLEAN |
| EBforUndetectableDangerRisk | √ | √ | REF BOOLEAN BOOLEAN |
| NotRestrictiveZoneLogicalInput | √ | √ | REF BOOLEAN BOOLEAN |
| PBforUndetectableDangerRisk | √ | √ | REF BOOLEAN BOOLEAN |
| PermissiveZoneLogicalInput | √ | √ | REF BOOLEAN BOOLEAN |
| UndetectDangerMotionWithoutNUDE | √ | √ | REF BOOLEAN BOOLEAN |
| UndetectableDangerRiskForNoNUDE | √ | √ | REF BOOLEAN BOOLEAN |

## F53-Monitor Train Speed

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| MemLocationNotConfirmed | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| MotionProtectioninhibition | Internal | REF \_Ref345055806 \h F46-Determine Over Energy |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainMaxSpeed | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| ValidTrainKinematic | Internal | REF \_Ref345055703 \h 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(i, k):
  
 return Offline.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(k):
  
 for i in range(0, MAX\_RM\_CONDITION\_NB):
  
 if (ConditionForRMlimitSpeed(i, k)):
  
 return 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设置为0，影响可用性。但对于这些限速的监控，仅在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(k):
  
 return (ValidTrainKinematic(k)
  
 and TrainMaxSpeed(k) &lt;= 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(k):
  
 return (not NoDangerforRMoverSpeed(k)
  
 and 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(k):
  
 return (not MemLocationNotConfirmed(k)
  
 or TrainMaxSpeed &lt; 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(k):
  
 return (not NoDangerforMemorizedLocationOverSpeed(k)
  
 and not 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 | √ | √ | REF BOOLEAN BOOLEAN |
| EBforMemorizedLocationOverSpeed | √ | √ | REF BOOLEAN BOOLEAN |
| EBforRMoverSpeed | √ | √ | REF BOOLEAN BOOLEAN |
| NoDangerForMemorizedLocationOverSpeed | √ | √ | REF BOOLEAN BOOLEAN |
| NoDangerForRMoverSpeed | √ | √ | REF BOOLEAN BOOLEAN |
| RMlimitSpeedApplied | √ | √ | REF BOOLEAN BOOLEAN |

## F54-Monitor Rollback Train Speed

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| End2RunningForward | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MaximumTrainMotion | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MinimumTrainMotion | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MotionProtectionInhibition | Internal | REF \_Ref345055806 \h F46-Determine Over Energy |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFrontEnd | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainMaxSpeed | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| ValidTrainKinematic | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| RollbackDistanceAccount\_1 | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| RollbackDistanceAccount\_2 | √ | √ | REF NUMERIC\_32 \h 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 represents the choice of RMR.

def RMRselectedDrivingMode(k):
  
 return Offline.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 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 in order to control that speed does not exceed ATPsetting.MPnotAuthLimitSpeed .

def RollbackDistanceAccount\_1(k):
  
 if (Initialization)
  
 return 0
  
 elif (not ValidTrainKinematic(k)):
  
 return (-1 \* ATPsetting.MPnotAuthDistWithoutMotionAvailable)
  
 elif (TrainFrontEnd(k) is END\_1
  
 and not RMRselectedDrivingMode(k)):
  
 if (OdometerState(k) is INITIALIZED):
  
 if (End1RunningForward(k)):
  
 return min(0, RollbackDistanceAccount\_1(k-1) + MinimumTrainMotion(k))
  
 else:
  
 return (RollbackDistanceAccount\_1(k-1) + MaximumTrainMotion(k))
  
 else:
  
 return RollbackDistanceAccount\_1(k-1)
  
 else:
  
 return 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 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 in order to control that speed does not exceed ATPsetting.MPnotAuthLimitSpeed.

def RollbackDistanceAccount\_2(k):
  
 if (Initialization)
  
 return 0
  
 elif (not ValidTrainKinematic(k)):
  
 return (-1 \* ATPsetting.MPnotAuthDistWithoutMotionAvailable)
  
 elif (TrainFrontEnd(k) is END\_2
  
 and not RMRselectedDrivingMode(k)):
  
 if (OdometerState(k) is INITIALIZED):
  
 if (End2RunningForward(k)):
  
 return min(0, RollbackDistanceAccount\_2(k-1) - MinimumTrainMotion(k))
  
 else:
  
 return (RollbackDistanceAccount\_2(k-1) â€” MaximumTrainMotion(k))
  
 else:
  
 return RollbackDistanceAccount\_2(k-1)
  
 else:
  
 return 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 shall initialize to False.
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 set as True, it shall stay at state True while ATP is not reboot.

def UnrecoverableRollbackOverSpeed(k):
  
 return (UnrecoverableRollbackOverSpeed(k-1)
  
 or (not RMRselectedDrivingMode(k)
  
 and ((TrainFrontEnd(k) is END\_2
  
 and (abs(RollbackDistanceAccount\_2(k))
  
 &gt; ATPsetting.MPnotAuthLimitDistance))
  
 or (TrainFrontEnd(k) is END\_1
  
 and (abs(RollbackDistanceAccount\_1(k))
  
 &gt; 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 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.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(k):
  
 return (not ValidTrainKinematic(k)
  
 or UnrecoverableRollbackOverSpeed(k)
  
 or (not RMRselectedDrivingMode(k)
  
 and (TrainMaxSpeed(k) &gt;= ATPsetting.MPnotAuthLimitSpeed
  
 and ((TrainFrontEnd(k) is END\_2
  
 and End1RunningForward(k) and not End2RunningForward(k))
  
 or (TrainFrontEnd is END\_1
  
 and End2RunningForward(k) and not 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 = 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 | √ | √ | REF BOOLEAN BOOLEAN |
| UnrecoverableRollbackOverSpeed | √ | √ | REF BOOLEAN BOOLEAN |
| RollbackOverSpeed | √ | √ | REF BOOLEAN BOOLEAN |
| EBforRollbackOverSpeed | √ | √ | REF BOOLEAN BOOLEAN |

## F55-Monitor Reverse Train Speed

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| End2RunningForward | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MaximumTrainMotion | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MinimumTrainMotion | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| MotionProtectionInhibition | Internal | REF \_Ref345055806 \h F46-Determine Over Energy |
| RMRselectedDrivingMode | Internal | REF \_Ref345055856 \h F54-Monitor Rollback Train Speed |
| TrainFrontEnd | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainMaxSpeed | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| ValidTrainKinematic | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| ReverseDistanceAccount\_1 | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| ReverseDistanceAccount\_2 | √ | √ | REF NUMERIC\_32 \h 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 represents the authorization of long distance reverse.

def LongDistanceReverseAuthorized(k):
  
 return Offline.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为True
在项目的安全配置参数中：
配置LongDistanceReverseAuthorized为上述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 511 Backward distance account rules

|  |  |  |
| --- | --- | --- |
| LongDistanceReverseAuthorized | not LongDistanceReverseAuthorized | 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 | 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 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 in order to control that speed does not exceed reverse speed limit function.

def ReverseDistanceAccount\_1(k):
  
 if (Initialization)
  
 return 0
  
 elif (not ValidTrainKinematic(k)):
  
 return ATPsetting.ReverseDistWithoutMotionAvailable
  
 elif (TrainFrontEnd is END\_1
  
 and not LongDistanceReverseAuthorized(k)):
  
 if (OdometerState(k) is INITIALIZED):
  
 if (End1RunningForward(k)):
  
 return min(0, ReverseDistanceAccount\_1(k-1) + MinimumTrainMotion(k))
  
 elif (RMRselectedDrivingMode(k)):
  
 return (ReverseDistanceAccount\_1(k-1) + MaximumTrainMotion(k))
  
 else:
  
 return ReverseDistanceAccount\_1(k-1)
  
 else:
  
 return ReverseDistanceAccount\_1(k-1)
  
 else:
  
 return 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 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 in order to control that speed does not exceed ReverseSpeedRestrictions reverse speed limit function.

def ReverseDistanceAccount\_2(k):
  
 if (Initialization)
  
 return 0
  
 elif (not ValidTrainKinematic(k)):
  
 return ATPsetting.ReverseDistWithoutMotionAvailable
  
 elif (TrainFrontEnd(k) is END\_2
  
 and not LongDistanceReverseAuthorized(k)):
  
 if (OdometerState(k) is INITIALIZED):
  
 if (End2RunningForward(k))
  
 return min(0, ReverseDistanceAccount\_2(k-1) - MinimumTrainMotion(k))
  
 elif (RMRselectedDrivingMode(k)):
  
 return (ReverseDistanceAccount\_2(k-1) â€” MaximumTrainMotion(k))
  
 else:
  
 return ReverseDistanceAccount\_2(k-1)
  
 else:
  
 return ReverseDistanceAccount\_2(k-1)
  
 else:
  
 return 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.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 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(k):
  
 if (not RMRselectedDrivingMode(k)
  
 or LongDistanceReverseAuthorized(k)):
  
 return False
  
 elif (not ValidTrainKinematic(k)):
  
 return True
  
 else:
  
 return ((TrainFrontEnd(k) is END\_2):
  
 and ((End1RunningForward(k) and not End2RunningForward(k)
  
 and (TrainMaxSpeed(k)
  
 &gt; ReverseSpeedRestrictions(ReverseDistanceAccount\_2(k))))
  
 or (ReverseSpeedRestrictions(ReverseDistanceAccount\_2(k)) == 0)))
  
 or (TrainFrontEnd(k) is END\_1
  
 and ((End2RunningForward(k) and not End1RunningForward(k)
  
 and (TrainMaxSpeed(k) &gt; ReverseSpeedRestrictions(ReverseDistanceAccount\_1(k))))
  
 or (ReverseSpeedRestrictions(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 = 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 | √ | √ | REF BOOLEAN BOOLEAN |
| ReverseOverSpeed | √ | √ | REF BOOLEAN BOOLEAN |
| ReverseSpeedRestrictions | √ | √ | REF NUMERIC\_32 \h 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 520 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 | Internal | REF \_Ref345055549 \h \* MERGEFORMAT F11-Acquire Configuration Data |
| ATPsetting | Internal | REF \_Ref345055549 \h \* MERGEFORMAT F11-Acquire Configuration Data |
| MotionProtectionInhibition | Internal | REF \_Ref345055806 \h \* MERGEFORMAT F46-Determine Over Energy |
| HYPERLINK \l &quot;NonVitalRequest&quot; NonVitalRequest | External | REF \_Ref348357813 \h \* MERGEFORMAT Interface with CC Non Vital |
| HYPERLINK \l &quot;Offline&quot; Offline | Internal | REF \_Ref345055549 \h \* MERGEFORMAT F11-Acquire Configuration Data |
| OtherATP | Internal | REF \_Ref345055575 \h \* MERGEFORMAT F13-Manage Loop Hour with Distant ATP |
| TrackMap | Internal | REF \_Ref345055549 \h \* MERGEFORMAT F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h \* MERGEFORMAT F27-Compute Train Kinematics |
| TrainFrontLocation | Internal | REF \_Ref345055747 \h \* MERGEFORMAT F33-Confirm Train Localization |
| TrainFrontOrientation | Internal | REF \_Ref345055747 \h \* MERGEFORMAT F33-Confirm Train Localization |
| TrainLocatedOnKnownPath | Internal | REF \_Ref345055747 \h \* MERGEFORMAT F33-Confirm Train Localization |
| TrainRearLocation | Internal | REF \_Ref345055747 \h \* MERGEFORMAT F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| NoVitalCorrectlyDocked | × | √ | REF BOOLEAN \* MERGEFORMAT BOOLEAN |
| TrainIncludedInVPEZ\_A | √ | √ | REF BOOLEAN \* MERGEFORMAT BOOLEAN |
| TrainIncludedInVPEZ\_B | √ | √ | REF BOOLEAN \* MERGEFORMAT BOOLEAN |
| TrainSafelyImmobilised | √ | √ | REF BOOLEAN \* MERGEFORMAT BOOLEAN |
| VPEZid\_A | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| VPEZid\_B | √ | √ | REF NUMERIC\_32 \h 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 512所示。
Table 512 Convertion from platform side to train side

|  |  |  |  |
| --- | --- | --- | --- |
| TrainFrontEnd | TrainFrontOrientation | Platform side in 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 |

[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中由一对方向相反的SGL\_VPEZ奇点组成）有交集，且根据Table 512判断为A侧时，设置TrainInterVPEZ\_A为True。
其他情况，设置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 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 512, set TrainInterVPEZ\_A as True:
Otherwise, set 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中由一对方向相反的SGL\_VPEZ奇点组成）有交集，且根据Table 512判断为B侧时，设置TrainInterVPEZ\_B为True。
其他情况，设置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 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 512, set TrainInterVPEZ\_B as True:
Otherwise, set 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为True，则记录该VPEZ的编号；
否则，设置为None。
VPEZid\_A records the ID of the VPEZ, which intersected with the location of train on side A.
If the 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为True，则记录该VPEZ的编号。；
否则，设置为None。
VPEZid\_B records the ID of the VPEZ, which intersected with the location of train on side B.
If the 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范围中，且TrainInterVPEZ\_A为True，则设置TrainIncludedInVPEZ\_A为True；
否则，设置TrainIncludedInVPEZ\_A为False.
TrainIncludedInVPEZ\_A, ATP shall determine whether the whole train location belongs to one vital passenger exchange zone VPEZid\_A.
In initialization or train delocalization, set the 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 is True, then ATP shall set TrainIncludedInVPEZ\_A as True;
Otherwise, set 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范围中，且该TrainInterVPEZ\_B为True，则设置TrainIncludedInVPEZ\_B为True；
否则，设置TrainIncludedInVPEZ\_B为False。
TrainIncludedInVPEZ\_B, ATP shall determine whether the whole train location belongs to one vital passenger exchange zone VPEZid\_B.
In initialization or train delocalization, set the 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 is True, then ATP shall set TrainIncludedInVPEZ\_B as True;
Otherwise, set 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 immobilizedATP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 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(k):
  
 return Offline.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 stands for that either of the train ends is in parking brake.

def TrainParkingBrakeApplied(k):
  
 return Offline.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(k):
  
 return ((TrainEmergencyBrakeApplied(k)
  
 or TrainParkingBrakeApplied(k))
  
 and 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(k):
  
 return (ATOcontrolTimeValid(k)
  
 and 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(k):
  
 return (TrainLocatedOnKnownPath(k)
  
 and TrainIncludedInVPEZ\_A(k)
  
 and NoVitalCorrectlyDocked(k)
  
 and 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(k):
  
 return (TrainLocatedOnKnownPath(k)
  
 and TrainIncludedInVPEZ\_B(k)
  
 and NoVitalCorrectlyDocked(k)
  
 and 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(k):
  
 return (LocalATPenableDoorOpening\_A(k)
  
 or 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(k):
  
 return (LocalATPenableDoorOpening\_B(k)
  
 or 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 | √ | √ | REF BOOLEAN BOOLEAN |
| EnableDoorOpening\_B | √ | √ | REF BOOLEAN BOOLEAN |
| LocalATPenableDoorOpening\_A | √ | √ | REF BOOLEAN BOOLEAN |
| LocalATPenableDoorOpening\_B | √ | √ | REF BOOLEAN BOOLEAN |
| TrainEmergencyBrakeApplied | √ | √ | REF BOOLEAN BOOLEAN |
| TrainInterVPEZ\_A | √ | √ | REF BOOLEAN BOOLEAN |
| TrainInterVPEZ\_B | √ | √ | REF BOOLEAN BOOLEAN |
| TrainPackingBrakeApplied | √ | √ | REF BOOLEAN 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 | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| NonVitalRequest | External | REF \_Ref348357813 \h Interface with CC Non Vital |
| OtherATP | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainFrontLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainLocatedOnKnownPath | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainRearLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| PSDmanagerOrder\_A | √ | √ | REF ST\_PSD\_MANAGE \h ST\_PSD\_MANAGE |
| PSDmanagerOrder\_B | √ | √ | REF ST\_PSD\_MANAGE \h ST\_PSD\_MANAGE |
| PSDoperation\_A | √ | √ | REF ST\_PSD\_OPERATION \h ST\_PSD\_OPERATION |
| PSDoperation\_B | √ | √ | REF ST\_PSD\_OPERATION \h ST\_PSD\_OPERATION |
| PSDzoneStatus\_A | √ | √ | REF ST\_PSD\_STATUS \h ST\_PSD\_STATUS |
| PSDzoneStatus\_B | √ | √ | REF ST\_PSD\_STATUS \h ST\_PSD\_STATUS |
| TableOfPSDPlatform | √ | × | REF ST\_PSD\_TABLE \h ST\_PSD\_TABLE |
| UsingPSDstatusFromCI | √ | √ | REF BOOLEAN 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为False，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中由一对方向相反的SGL\_PSD\_ZONE奇点组成）有交集，且根据Table 512判断为A侧时，设置AlignPSDzone\_A为True，并将PSDid\_A为设置为该SGL\_PSD\_ZONE奇点的id。
其他情况，设置AlignPSDzone\_A为False，PSDid\_A为None。
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, the id of the PSD on side-A intersects with train location.
In initialization or train delocalization, set AlignPSDzone\_A as False and 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 512, set AlignPSDzone\_A as True, and records PSDid\_A as the PSD’s id:
Otherwise, set AlignPSDzone\_A as False and 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为False，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中由一对方向相反的SGL\_PSD\_ZONE奇点组成）有交集，且根据Table 512判断为B侧时，设置AlignPSDzone\_B为True，并将PSDid\_B为设置为该SGL\_PSD\_ZONE奇点的id。
其他情况，设置AlignPSDzone\_B为False，PSDid\_B为None。
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, the id of the PSD on side-A intersects with train location.
In initialization or train delocalization, set AlignPSDzone\_B as False and 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 512, set AlignPSDzone\_B as True, and records PSDid\_B as the PSD’s id:
Otherwise, set AlignPSDzone\_B as False and 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 and PSDoperation\_B structured as ST\_PSD\_OPERATION, used to obtain the PSD controlling order from CCNV.

if (ATOcontrolTimeValid(k) == True)
  
 PSDoperation\_A(k)= NonVitalRequest.PSDoperation\_A(k)
  
 PSDoperation\_B(k)= NonVitalRequest.PSDoperation\_B(k)
  
 else:
  
 PSDoperation\_A(k).Id = None
  
 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或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(k):
  
 return (PSDoperation\_A(k).Id is not None
  
 or 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 | REF NUMERIC\_32 \h NUMERIC\_32 | PSD状态是否有效 | DoorClosed |

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消息时，根据其ID号，更新TableOfPSDPlatform数组中相应PSD的DoorClosed状态，并将DoorStatusValidityTime设为ATPsetting.PSDstatusValidityTime减去FSFB2消息传输延迟。
In initialization, all PSD DoorClosed of TableOfPSDPlatform set as False, and the valid period is zero. Later on, if receiving correct CI\_IOstatus, on the basis of ID number the status of related PSD doorClosed from TableOfPSDPlatform is updated and the DoorStatusValidityTime is set as 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消息，或者该消息校验错误，则ATP更新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 from CI, or if this message detected as False, ATP shall update the PSD status of 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; 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 and PSDzoneStatus\_B based on the whether ATP is master or not, the information from interlock device, ZC or BM.

|  |  |  |  |
| --- | --- | --- | --- |
| Identification | Logical Type | Description | ST\_PSD\_STATUS |
| Id | REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | PSD的标识 | Validity |
| REF NUMERIC\_32 \h \* MERGEFORMAT NUMERIC\_32 | PSD剩余有效期 | AllPSDclosed | REF BOOLEAN \h \* MERGEFORMAT BOOLEAN |

[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 = ((AlignPSDzone\_A(k) or AlignPSDzone\_B(k))
  
 and (TrainFilteredStopped(k)
  
 or (TrainFilteredStopped(k-1)
  
 and not 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 shows whether the status from CCNV is the main controlled CC.

if (ATOcontrolTimeValid(k) == True)
  
 MasterCCcore = NonVitalRequest.MasterCcCore(k)
  
 else:
  
 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, 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(k) == True)
  
 if ((PSDoperation\_A.Id == None) or (PSDoperation\_A.Id != PSDid\_A(k)))
  
 PSDzoneStatus\_A.Id(k)= None
  
 PSDzoneStatus\_A.Validity = 0
  
 PSDzoneStatus\_A.AllPSDclosed = False
  
 else:
  
 PSDzoneStatus\_A(k).Id = PSDoperation\_A(k).Id
  
 if (UsingPSDstatusFromCI(k))
  
 PSDzoneStatus\_A.Validity(k) = TableOfPSDPlatform[PSDid\_A(k)].DoorStatusValidityTime
  
 PSDzoneStatus\_A.AllPSDclosed(k) = TableOfPSDPlatform[PSDid\_A(k)].DoorClosed
  
 elif (CoercedPermissive(TrackMap.PSDs[PSDid\_A(k)].CoercedPermissive, k))
  
 PSDzoneStatus\_A.Validity = REPORT\_AGE\_MAX
  
 PSDzoneStatus\_A.AllPSDclosed = True
  
 elif (CoercedRestrictive(TrackMap.PSDs[PSDid\_A(k)].NotCoercedRestrictive, k))
  
 PSDzoneStatus\_A.Validity = REPORT\_AGE\_MAX
  
 PSDzoneStatus\_A.AllPSDclosed = False
  
 else:
  
 PSDzoneStatus\_A.Validity(k) = ReceivedVariantReport[LineSectionOfPSD].ValidityTime
  
 PSDzoneStatus\_A.AllPSDclosed(k)= VariantValue(TrackMap.PSDs[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, 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(k) == True)
  
 if ((PSDoperation\_B.Id == None) or (PSDoperation\_B.Id != PSDid\_B(k)))
  
 PSDzoneStatus\_B.Id(k)= None
  
 PSDzoneStatus\_B.Validity = 0
  
 PSDzoneStatus\_B.AllPSDclosed = False
  
 else:
  
 PSDzoneStatus\_B.Id(k)= PSDoperation\_B.Id
  
 if (UsingPSDstatusFromCI(k))
  
 PSDzoneStatus\_B.Validity(k) = TableOfPSDPlatform[PSDid\_B(k)].DoorStatusValidityTime
  
 PSDzoneStatus\_B.AllPSDclosed(k) = TableOfPSDPlatform[PSDid\_B(k)].DoorClosed
  
 elif (CoercedPermissive(TrackMap.PSDs[PSDid\_B(k)].CoercedPermissive, k))
  
 PSDzoneStatus\_B.Validity = REPORT\_AGE\_MAX
  
 PSDzoneStatus\_B.AllPSDclosed = True
  
 elif (CoercedRestrictive(TrackMap.PSDs[PSDid\_B(k)].NotCoercedRestrictive, k))
  
 PSDzoneStatus\_B.Validity = REPORT\_AGE\_MAX
  
 PSDzoneStatus\_B.AllPSDclosed = False
  
 else:
  
 PSDzoneStatus\_B.Validity(k) = ReceivedVariantReport[LineSectionOfPSD].ValidityTime
  
 PSDzoneStatus\_B.AllPSDclosed(k)= VariantValue(TrackMap.PSDs[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 != True)
  
 if (OtherATPmessageAvailable(k) == True)
  
 PSDzoneStatus\_A.Id = OtherATP.PsdIdSide\_A
  
 if (PSDzoneStatus\_A.Id(k) != None)
  
 PSDzoneStatus\_A.Validity(k) = (OtherATP.PsdValiditySide\_A
  
 - Message.ModularSub(ATPtime(k), OtherATP.LatestTimeOtherCore))
  
 PSDzoneStatus\_A.AllPSDclosed(k) = (OtherATP.PsdClosedSide\_A
  
 and (PSDzoneStatus\_A.Validity(k) &gt; 0))
  
 else:
  
 PSDzoneStatus\_A.Validity = 0
  
 PSDzoneStatus\_A.AllPSDclosed = False
  
 elif (PSDzoneStatus\_A.Id(k-1) != None)
  
 PSDzoneStatus\_A.Id = PSDzoneStatus\_A.Id(k-1)
  
 PSDzoneStatus\_A.Validity = PSDzoneStatus\_A.Validity(k-1)- 1
  
 PSDzoneStatus\_A.AllPSDclosed(k) = (PSDzoneStatus\_A.AllPSDclosed(k-1)
  
 and (PSDzoneStatus\_A.Validity(k) &gt; 0))
  
 else:
  
 PSDzoneStatus\_A.Validity = 0
  
 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 != True)
  
 if (OtherATPmessageAvailable(k) == True)
  
 PSDzoneStatus\_B.Id = OtherATP.PsdIdSide\_B
  
 if (PSDzoneStatus\_B.Id(k) != None)
  
 PSDzoneStatus\_B.Validity(k) = (OtherATP.PsdValiditySide\_B
  
 - Message.ModularSub(ATPtime(k), OtherATP.LatestTimeOtherCore))
  
 PSDzoneStatus\_B.AllPSDclosed(k)
  
 = OtherATP.PsdClosedSide\_B
  
 and (PSDzoneStatus\_B.Validity(k) &gt; 0)
  
 else:
  
 PSDzoneStatus\_B.Validity = 0
  
 PSDzoneStatus\_B.AllPSDclosed = False
  
 elif (PSDzoneStatus\_B.Id(k-1) != None)
  
 PSDzoneStatus\_B.Id = PSDzoneStatus\_B.Id(k-1)
  
 PSDzoneStatus\_B.Validity = PSDzoneStatus\_B.Validity(k-1) - 1
  
 PSDzoneStatus\_B.AllPSDclosed(k)
  
 = PSDzoneStatus\_B.AllPSDclosed(k-1)
  
 and (PSDzoneStatus\_B.Validity(k) &gt; 0)
  
 else:
  
 PSDzoneStatus\_B.Validity = 0
  
 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(k):
  
 if (PSDzoneStatus\_A(k).Id is None
  
 or PSDzoneStatus\_A(k).Validity &lt;= 0):
  
 return PSD\_STATE\_UNKNOWN
  
 elif (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(k):
  
 if (PSDzoneStatus\_B(k).Id is None
  
 or PSDzoneStatus\_B(k).Validity &lt;= 0):
  
 return PSD\_STATE\_UNKNOWN
  
 elif (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(k):
  
 PSDmanagerOrder\_A.Id = PSDoperation\_A.Id(k)
  
 if (PSDoperation\_A.Id(k) == PSDid\_A(k)
  
 and PSDoperation\_A.Id(k) is not None
  
 and not PSDoperation\_A.ClosingOrder(k)
  
 and PSDoperation\_A.OpeningOrder(k)
  
 and EnableDoorOpening\_A(k)):
  
 PSDmanagerOrder\_A.Order = Open\_PSD\_Configuration
  
 elif (not PSDoperation\_A.OpeningOrder(k)
  
 and PSDoperation\_A.ClosingOrder(k)):
  
 PSDmanagerOrder\_A.Order = Close\_PSD\_Of\_Platform
  
 else:
  
 PSDmanagerOrder\_A.Order = None
  
 return 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(k):
  
 PSDmanagerOrder\_B.Id = PSDoperation\_B.Id(k)
  
 if (PSDoperation\_B.Id(k) == PSDid\_B(k)
  
 and PSDoperation\_B.Id(k) is not None
  
 and not PSDoperation\_B.ClosingOrder(k)
  
 and PSDoperation\_B.OpeningOrder(k)
  
 and EnableDoorOpening\_B(k)):
  
 PSDmanagerOrder\_B.Order = Open\_PSD\_Configuration
  
 elif (not PSDoperation\_B.OpeningOrder(k)
  
 and PSDoperation\_B.ClosingOrder(k)):
  
 PSDmanagerOrder\_B.Order = Close\_PSD\_Of\_Platform
  
 else:
  
 PSDmanagerOrder\_B.Order = None
  
 return 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 = False
  
 elif ((PSDmanagerOrder\_A.Order(k) == Open\_PSD\_Configuration)
  
 or (PSDmanagerOrder\_B.Order(k) == Open\_PSD\_Configuration))
  
 PSDplatformManagerOpeningOrder = True
  
 else:
  
 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(k) == True)
  
 or (OtherATP.PsdManagerOpeningOrder(k) == True))
  
 PSDopeningCommand = True
  
 else:
  
 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.Id有效，则根据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[0]。
When communicating with the CI, if the PSDmanagerOrder\_A which comes from CCNV was valid, ATP shall set the CIsetting[0] according to the PSDmanagerOrder\_A and the configuration of the PSD.

if ((CommunicateWithPSD(k) == True)
  
 and (PSDmanagerOrder\_A.Id != None))
  
 CIsetting[0].PlatformId = PSDmanagerOrder\_A.Id
  
 if (PSDmanagerOrder\_A.Order == Open\_PSD\_Configuration)
  
 CIsetting[0].Order = TrackMap.PSDs[PSDmanagerOrder\_A.Id].DoorOpeningCode
  
 elif (PSDmanagerOrder\_A.Order == Close\_PSD\_Configuration):
  
 CIsetting[0].Order = TrackMap.PSDs[PSDmanagerOrder\_A.Id].DoorClosingCode
  
 else:
  
 CIsetting[0].Order = TrackMap.PSDs[PSDmanagerOrder\_A.Id].DoorNoActionCode
  
 else:
  
 CIsetting[0].PlatformId = None
  
 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.Id有效，则根据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 [1]。
When communicating with the CI, if the PSDmanagerOrder\_B, which comes from CCNV, was valid, ATP shall set the CIsetting[1] according to the PSDmanagerOrder\_B and the configuration of the PSD.

if ((CommunicateWithPSD(k) == True)
  
 and (PSDmanagerOrder\_B.Id != None)):
  
 CIsetting[1].PlatformId = PSDmanagerOrder\_B.Id
  
 if (PSDmanagerOrder\_B.Order == Open\_PSD\_Configuration)
  
 CIsetting[1].Order = TrackMap.PSDs[PSDmanagerOrder\_B.Id].DoorOpeningCode
  
 elif (PSDmanagerOrder\_B.Order == Close\_PSD\_Configuration):
  
 CIsetting[1].Order = TrackMap.PSDs[PSDmanagerOrder\_B.Id].DoorClosingCode
  
 else:
  
 CIsetting[1].Order = TrackMap.PSDs[PSDmanagerOrder\_B.Id].DoorNoActionCode
  
 else:
  
 CIsetting[1].PlatformId = None
  
 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 | √ | √ | REF BOOLEAN BOOLEAN |
| AlignPSDzone\_B | √ | √ | REF BOOLEAN BOOLEAN |
| CommunicateWithPSD | √ | √ | REF BOOLEAN BOOLEAN |
| MasterCCcore | × | √ | REF BOOLEAN BOOLEAN |
| PSDid\_A | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| PSDid\_B | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| PSDopeningCommand | √ | √ | REF BOOLEAN BOOLEAN |
| PSDplatformManagerOpeningOrder | √ | √ | REF BOOLEAN BOOLEAN |
| PSDstatusNonVital\_A | × | √ | REF ENUM\_PSD\_STATE \h ENUM\_PSD\_STATE |
| PSDstatusNonVital\_B | × | √ | REF ENUM\_PSD\_STATE \h 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 | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| AlignPSDzone\_A | Internal | REF \_Ref345055899 \h F62-Control PSD Opening and Closing Order |
| AlignPSDzone\_B | Internal | REF \_Ref345055899 \h F62-Control PSD Opening and Closing Order |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainInterVPEZ\_A | Internal | REF \_Ref268160839 \h F61-Elaborate Door Opening Authorization |
| TrainInterVPEZ\_B | Internal | REF \_Ref268160839 \h F61-Elaborate Door Opening Authorization |
| TrainLocalized | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainParkingBrakeApplied | Internal | REF \_Ref268160839 \h F61-Elaborate Door Opening Authorization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| AllPSDclosedAndLocked | √ | √ | REF BOOLEAN BOOLEAN |
| AllTrainDoorsClosedAndLocked | √ | √ | REF BOOLEAN BOOLEAN |
| PSDopeningCommandValid | √ | √ | REF BOOLEAN 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 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(k):
  
 return Offline.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(k):
  
 return Offline.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(k):
  
 return not (TrainFilteredStopped(k)
  
 and not AllTrainDoorsClosedAndLocked(k)
  
 and (AlignPSDzone\_A(k) or AlignPSDzone\_B(k)
  
 or TrainInterVPEZ\_A(k) or 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(k):
  
 return (not NoDangerForTrainDoorsNotClosedAndLocked(k)
  
 and not 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(k)
  
 = ((PBforTrainDoorsNotClosedAndLocked(k) == True)
  
 and (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(k):
  
 return (AllTrainDoorsClosedAndLocked(k)
  
 or TrainFilteredStopped(k)
  
 or not 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(k):
  
 return (not NoDangerForDepartureWithoutTDCL(k)
  
 and not 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和车门状态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 513 Configuration for monitoring unexpected train door open

|  |  |  |
| --- | --- | --- |
| Project configuration | With REF \_Ref385344693 \h Door opening authorization | Without REF \_Ref385344693 \h 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 |

[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 is selected.

def InhibitProtectionMovingWithoutTDCL(k):
  
 return Offline.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(k):
  
 return (AllTrainDoorsClosedAndLocked(k)
  
 or TrainFilteredStopped(k)
  
 or EnableDoorOpening\_A(k)
  
 or 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(k):
  
 return (not NoDangerForMovingWithoutTDCL(k)
  
 and not 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(k):
  
 return Offline.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 is True.

AllPSDclosedAndLocked(k)
  
 = ((TrainLocalized(k) == True)
  
 and ((TrainFilteredStopped(k) == True)
  
 or (TrainFilteredStopped(k-1) == True))
  
 and ((PSDid\_A(k) == 0) and (PSDid\_B(k) == 0))
  
 or ((PSDid\_A(k) != 0) and (PSDzoneStatus\_A.AllPSDclosed(k) == True)
  
 and (PSDid\_B(k) == 0))
  
 or ((PSDid\_B(k) != 0) and (PSDzoneStatus\_B.AllPSDclosed(k) == True)
  
 and (PSDid\_A(k) == 0))
  
 or ((PSDid\_A(k) != 0) and (PSDzoneStatus\_A.AllPSDclosed(k) == True)
  
 and (PSDid\_B(k) != 0) and (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(k):
  
 return (not (AlignPSDzone\_A(k) or AlignPSDzone\_B(k))
  
 or AllPSDclosedAndLocked(k)
  
 or TrainFilteredStopped(k)
  
 or not 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(k):
  
 return (not NoDangerForUnexpectedPSDopening(k)
  
 and not 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(k):
  
 return not (TrainFilteredStopped(k)
  
 and not AllPSDclosedAndLocked(k)
  
 and (AlignPSDzone\_A(k) or 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(k):
  
 return (not NoDangerForPSDnotClosedAndLocked(k)
  
 and not 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(k)
  
 = ((PBforPSDnotClosedAndLocked(k) == True)
  
 and (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为True，则设置本周期PSDopeningCommandValid为True；
否则如果上周期PSDopeningCommand为True而本周期变为False，则在接下来的ATPsetting. PSDopeningCommandValidityTime周期内：
如果PSDopeningCommand为False，仍保持PSDopeningCommandValid为True；
如果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.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(k):
  
 return Offline.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(k):
  
 return (PSDopeningCommandValid(k)
  
 and not 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 | √ | √ | REF BOOLEAN BOOLEAN |
| EBforMovingWithoutTDCL | √ | √ | REF BOOLEAN BOOLEAN |
| EBforPBnotAppliedDueToPSD | √ | √ | REF BOOLEAN BOOLEAN |
| EBforPBnotAppliedDueToTrainDoors | √ | √ | REF BOOLEAN BOOLEAN |
| EBforUnexpectedPSDopening | √ | √ | REF BOOLEAN BOOLEAN |
| InhibitControlPSDstatus | √ | √ | REF BOOLEAN BOOLEAN |
| InhibitControlTrainDoorsStatus | √ | √ | REF BOOLEAN BOOLEAN |
| InhibitProtectionMovingWithoutTDCL | √ | √ | REF BOOLEAN BOOLEAN |
| InhibitPSDopeningSupervisedByATP | √ | √ | REF BOOLEAN BOOLEAN |
| NoDangerForDepartureWithoutTDCL | √ | √ | REF BOOLEAN BOOLEAN |
| NoDangerForMovingWithoutTDCL | √ | √ | REF BOOLEAN BOOLEAN |
| NoDangerForPSDnotClosedAndLocked | √ | √ | REF BOOLEAN BOOLEAN |
| NoDangerForTrainDoorsNotClosedAndLocked | √ | √ | REF BOOLEAN BOOLEAN |
| NoDangerForUnexpectedPSDopening | √ | √ | REF BOOLEAN BOOLEAN |
| PBforPSDnotClosedAndLocked | √ | √ | REF BOOLEAN BOOLEAN |
| PBforPSDopenedAndSupervisedByATP | √ | √ | REF BOOLEAN BOOLEAN |
| PBforTrainDoorsNotClosedAndLocked | √ | √ | REF BOOLEAN 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 | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| EnableDoorOpening\_A | Internal | REF \_Ref268160839 \h F61-Elaborate Door Opening Authorization |
| EnableDoorOpening\_B | Internal | REF \_Ref268160839 \h F61-Elaborate Door Opening Authorization |
| InhibitEmergencyBrake | Internal | REF \_Ref345055931 \h F71-Outputs to Rolling-stock |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainMaxSpeed | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| ValidTrainKinematic | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| EvacuationNotPossible\_A | √ | √ | REF BOOLEAN BOOLEAN |
| EvacuationNotPossible\_B | √ | √ | REF BOOLEAN BOOLEAN |
| TrainDockedInStaion | √ | √ | REF BOOLEAN BOOLEAN |
| TrainLeavingStation | √ | √ | REF BOOLEAN 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 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(k):
  
 return Offline.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 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(k):
  
 return 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(k)
  
 = ((ValidTrainKinematic(k) and (not TrainFilteredStopped(k)))
  
 or 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(k)
  
 = ((ValidTrainKinematic(k) and (not TrainFilteredStopped(k)))
  
 or 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(k)
  
 = (not HoldDoorsClosedTrainEnd1(k)
  
 or not 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(k):
  
 return (HoldDoorsClosedTrainEnd1(k)
  
 or 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(k):
  
 return (HoldDoorsClosedTrainEnd2(k)
  
 or 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为False时，默认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 512判断为A侧时，设置EvacuationNotPossible\_A为True：
其他情况，设置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, set the 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 512, set EvacuationNotPossible\_A as True.
Otherwise, set 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为False时，默认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 512判断为B侧时，设置EvacuationNotPossible\_B为True。
其他情况，设置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, set the 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 512, set EvacuationNotPossible\_B as True.
Otherwise, set 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 show as following ARDL:

if (Initialization
  
 or (ValidTrainKinematic(k) != True))
  
 HoldDoorsClosed\_A = False
  
 elif (TrainMaxSpeed(k) &gt; ATPsetting.DoorTrainLockingSpeed)
  
 HoldDoorsClosed\_A = True
  
 elif (TrainMaxSpeed(k) &lt;= ATPsetting.DoorTrainUnlockingSpeed)
  
 HoldDoorsClosed\_A(k)
  
 = ((EvacuationNotPossible\_A(k) and (not EvacuationNotPossible\_B(k)))
  
 or (EvacuationNotPossible\_A(k) and EvacuationNotPossible\_B(k)
  
 and ATPsetting.EvacuationTrainEnd))
  
 else:
  
 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 show as following ARDL :

if (Initialization
  
 or (ValidTrainKinematic(k) != True))
  
 HoldDoorsClosed\_B = False
  
 elif (TrainMaxSpeed(k) &gt; ATPsetting.DoorTrainLockingSpeed)
  
 HoldDoorsClosed\_B = True
  
 elif (TrainMaxSpeed(k) &lt;= ATPsetting.DoorTrainUnlockingSpeed)
  
 HoldDoorsClosed\_B(k)
  
 = ((EvacuationNotPossible\_B(k) and (not EvacuationNotPossible\_A(k)))
  
 or (EvacuationNotPossible\_A(k) and EvacuationNotPossible\_B(k)
  
 and ATPsetting.EvacuationTrainEnd))
  
 else:
  
 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(k):
  
 return (TrainFilteredStopped(k)
  
 and (EnableDoorOpening\_A(k)
  
 or 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由True变为False开始，如果列车测速有效，累加MaximumTrainMotion距离：
如果其绝对值在[0, 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), the cumulated of the absolute value of MaximumTrainMotion is in the range [0, ATPsetting.EvacuationStationAreaLength] and no train kinematic invalidation occurs.
or else, if the train kinematics is invalid, ATP shall set TrainLeavingStation as False and clear the cumulated distance.

def TrainLeavingStation(k):
  
 if (Initialization
  
 or not ValidTrainKinematic(k)):
  
 TrainHasDockedInStation = False
  
 LeavingStationDistance = 0
  
 return False
  
 elif (TrainDockedInStation(k)):
  
 TrainHasDockedInStation = True
  
 LeavingStationDistance = 0
  
 return False
  
 elif (not TrainHasDockedInStation(k)):
  
 LeavingStationDistance = 0
  
 return False
  
 else:
  
 LeavingStationDistance = LeavingStationDistance(k-1) + MaximumTrainMotion(k)
  
 return (abs(LeavingStationDistance) &lt;= 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(k):
  
 return Offline.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(k)
  
 = ((EmergencyHandleNotPulledSide(k) != True)
  
 and (TrainLeavingStation(k) == True)
  
 and (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(k)
  
 = ((EmergencyHandleNotPulledSide(k) != True)
  
 and (TrainFilteredStopped(k) == True)
  
 and not EnableDoorOpening\_A(k)
  
 and not 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(k):
  
 return Offline.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(k):
  
 return Offline.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(k):
  
 return (EvacuationWhileLeavingStation(k)
  
 and not 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(k):
  
 return (EvacuationWithTrainStopped(k)
  
 and not 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 | √ | √ | REF BOOLEAN BOOLEAN |
| EBforEvacuationWithTrainStopped | √ | √ | REF BOOLEAN BOOLEAN |
| EBforNotAllTrainEndHoldDoorsClosed | √ | √ | REF BOOLEAN BOOLEAN |
| EmergencyDetrainDoorLockingEnd1 | √ | √ | REF BOOLEAN BOOLEAN |
| EmergencyDetrainDoorLockingEnd2 | √ | √ | REF BOOLEAN BOOLEAN |
| EmergencyHandleNotPulledSide | √ | √ | REF BOOLEAN BOOLEAN |
| EmergencyHandleNotPulledEnd1 | √ | √ | REF BOOLEAN BOOLEAN |
| EmergencyHandleNotPulledEnd2 | √ | √ | REF BOOLEAN BOOLEAN |
| EvacuationWhileLeavingStation | √ | √ | REF BOOLEAN BOOLEAN |
| EvacuationWithTrainStopped | √ | √ | REF BOOLEAN BOOLEAN |
| HoldDoorsClosedTrainEnd1 | √ | √ | REF BOOLEAN BOOLEAN |
| HoldDoorsClosedTrainEnd2 | √ | √ | REF BOOLEAN BOOLEAN |
| HoldDoorsClosed\_A | √ | √ | REF BOOLEAN BOOLEAN |
| HoldDoorsClosed\_B | √ | √ | REF BOOLEAN BOOLEAN |
| InhibitProtectionEvacuationInDistance | √ | √ | REF BOOLEAN BOOLEAN |
| InhibitProtectionEvacuationWithStop | √ | √ | REF BOOLEAN BOOLEAN |

## F7-Generate Output Orders

Figure 521 SART modeling of function F7

## F71-Outputs to Rolling-stock

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATOcontrolTimeValid | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPtime | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| CoreId | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| EBforDepartureWithoutTDCL | Internal | REF \_Ref278299667 \h F63-Monitor Status of Doors |
| EBforEvacuationWhileTrainLeavingStation | Internal | REF \_Ref268160951 \h \* MERGEFORMAT F64-Protect Passengers during Emergency Evacuation |
| EBforEvacuationWithTrainStopped | Internal | REF \_Ref268160951 \h \* MERGEFORMAT F64-Protect Passengers during Emergency Evacuation |
| EBforMemorizedLocationOverSpeed | Internal | REF \_Ref345055848 \h F53-Monitor Train Speed |
| EBforMovingWithoutTDCL | Internal | REF \_Ref278299667 \h F63-Monitor Status of Doors |
| EBforNotAllTrainEndHoldDoorsClosed | Internal | REF \_Ref268160951 \h \* MERGEFORMAT F64-Protect Passengers during Emergency Evacuation |
| EBforOverEnergy | Internal | REF \_Ref345055806 \h F46-Determine Over Energy |
| EBforPBnotAppliedDueToPSD | Internal | REF \_Ref278299667 \h F63-Monitor Status of Doors |
| EBforPBnotAppliedDueToTrainDoors | Internal | REF \_Ref278299667 \h F63-Monitor Status of Doors |
| EBforReverseOverSpeed | Internal | REF \_Ref345055863 \h F55-Monitor Reverse Train Speed |
| EBforRMoverSpeed | Internal | REF \_Ref345055848 \h F53-Monitor Train Speed |
| EBforRollbackOverSpeed | Internal | REF \_Ref345055856 \h F54-Monitor Rollback Train Speed |
| EBforUnexpectedPSDopening | Internal | REF \_Ref278299667 \h F63-Monitor Status of Doors |
| EBforUndetectableDangerRisk | Internal | REF \_Ref345055837 \h F52-Prevent Train Moving in Undetectable Danger |
| EBonNonExclusiveRoute | Internal | REF \_Ref345055830 \h F51-Moral Time |
| LatestVIOM1LoopHourVIOM | Internal | REF \_Ref345055566 \h \* MERGEFORMAT F12-Manage Train Status |
| LatestVIOM2LoopHourVIOM | Internal | REF \_Ref345055566 \h \* MERGEFORMAT F12-Manage Train Status |
| MasterCCcore | Internal | REF \_Ref345055899 \h F62-Control PSD Opening and Closing Order |
| MatchRebootCondition | Internal | REF \_Ref345055566 \h F12-Manage Train Status |
| NonVitalRequest | External | REF \_Ref348357813 \h Interface with CC Non Vital |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| OtherATP | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| OtherATPmessageValid | Internal | REF \_Ref345055575 \h F13-Manage Loop Hour with Distant ATP |
| OtherCoreId | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| PBforOverEnergy | Internal | REF \_Ref345055806 \h F46-Determine Over Energy |
| PBforPSDnotClosedAndLocked | Internal | REF \_Ref278299667 \h F63-Monitor Status of Doors |
| PBforPSDopenedAndSupervisedByATP | Internal | REF \_Ref278299667 \h F63-Monitor Status of Doors |
| PBforTrainDoorsNotClosedAndLocked | Internal | REF \_Ref278299667 \h F63-Monitor Status of Doors |
| PBforUndetectableDangerRisk | Internal | REF \_Ref345055837 \h F52-Prevent Train Moving in Undetectable Danger |
| PBonNonExclusiveRoute | Internal | REF \_Ref345055830 \h F51-Moral Time |
| SafeTimerFailed | Internal | REF \_Ref345055971 \h F82-Manage Vital Time |
| SubSystemId | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| ApproachableSignalOverrun | √ | √ | REF BOOLEAN BOOLEAN |
| CCworkOvertime | √ | √ | REF BOOLEAN BOOLEAN |
| EBforOperationRequest | √ | √ | REF BOOLEAN BOOLEAN |
| IncompatibleDistantATP | √ | √ | REF BOOLEAN BOOLEAN |
| PBforOperationRequest | √ | √ | REF BOOLEAN BOOLEAN |
| TrainEmergencyBrakeRequested | √ | √ | REF BOOLEAN BOOLEAN |
| TrainParkingBrakeRequested | √ | √ | REF BOOLEAN BOOLEAN |
| VIOM1OutNotDisabled | √ | √ | REF BOOLEAN BOOLEAN |
| VIOM2OutNotDisabled | √ | √ | REF BOOLEAN BOOLEAN |

### Processing

Parking brakeATP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(k) == True)
  
 PBforOperationalRequest(k)
  
 = not NonVitalRequest.VitalParkingBrakingNotRequested(k)
  
 else:
  
 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, 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 = (PBonNonExclusiveRoute(k)
  
 or PBforOverEnergy(k)
  
 or PBforTrainDoorsNotClosedAndLocked(k)
  
 or PBforPSDnotClosedAndLocked(k)
  
 or PBforUndetectableDangerRisk(k)
  
 or PBforOperationalRequest(k)
  
 or PBforPSDopenedAndSupervisedByATP(k)
  
 or not 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，ATP software do not apply the parking brake.

InhibitParkingBrake = not TrainParkingBrakeRequested(k)

#Category=Functional
#Contribution=SIL4
#Allocation=ATP Software
#Source=[iTC\_CC-SyAD-0230]
[End]

Emergency brakeATP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和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 and SubSystemID of the ATP and redundant ATP need to compare for the consistency, which records in 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(k):
  
 if (Initialization
  
 or not OtherATPmessageValid(k)):
  
 return False
  
 elif (OtherCoreId(k) != OtherATP(k).CoreId
  
 or SubSystemId(k) != OtherATP(k).CC\_SSID):
  
 return True
  
 else:
  
 return 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；
上周期ApproachableSignalOverrun为False；
本周期列车位移MaximumTrainMotion向激活的驾驶室方向运行；
本周期列车车头最大定位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 59）。
否则，设置ApproachableSignalOverrun为False。
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 as True.
not MotionProtectionInhibition;
And 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 59).
Otherwise, ATP set 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(k) == True)
  
 EBforOperationalRequest(k) = not NonVitalRequest.EmergencyBrakingNotRequested(k)
  
 else:
  
 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 &quot;all PSD proven closed and locked&quot;;
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(k)
  
 = EBonNonExclusiveRoute(k)
  
 or EBforOverEnergy(k)
  
 or EBforRMoverSpeed(k)
  
 or EBforMemorizedLocationOverSpeed(k)
  
 or EBforRollbackOverSpeed(k)
  
 or EBforReverseOverSpeed(k)
  
 or EBforEvacuationWhileTrainLeavingStation(k)
  
 or EBforEvacuationWithTrainStopped(k)
  
 or EBforDepartureWithoutTDCL(k)
  
 or EBforMovingWithoutTDCL(k)
  
 or EBforUnexpectedPSDopening(k)
  
 or EBforUndetectableDangerRisk(k)
  
 or EBforOperationalRequest(k)
  
 or EBforNotAllTrainEndHoldDoorsClosed(k)
  
 or EBforPBnotAppliedDueToTrainDoors(k)
  
 or EBforPBnotAppliedDueToPSD(k)
  
 or ApproachableSignalOverrun(k)
  
 or SafeTimerFailed(k)
  
 or 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(k-1) == True)
  
 InhibitEmergencyBrake = not TrainEmergencyBrakeRequested(k)
  
 elif ((InhibitEmergencyBrake(k-1) == False)
  
 and (TrainFilteredStopped(k) == True))
  
 InhibitEmergencyBrake = not TrainEmergencyBrakeRequested(k)
  
 else:
  
 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(k):
  
 return ((CoreId(k) is END\_1
  
 and ((ATPtime(k) - CC1\_INIT\_TIME) &gt; MAX\_RESET\_TIME))
  
 or (CoreId(k) is END\_2
  
 and ((ATPtime(k) - CC2\_INIT\_TIME) &gt; 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(port, k):
  
 return (ATOcontrolTimeValid(k)
  
 and NonVitalRequest(k).Viom1[port])

def VIOM2OutNotDisabled(port, k):
  
 return (ATOcontrolTimeValid(k)
  
 and 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 &quot;not disabled&quot; 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 514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(k):
  
 for port in range(0, MAX\_VITAL\_OUTPUT\_NB):
  
 if (not CCworkOvertime(k)
  
 and not MatchRebootCondition(k)
  
 and VIOM1OutNotDisabled(k)[port]):
  
 VIOM1VitalOut[port] = Offline.GetVIOM1VitalOut(port)
  
 else:
  
 VIOM1VitalOut[port] = False
  
 return VIOM1VitalOut

def VIOM2VitalOut(k):
  
 for port in range(0, MAX\_VITAL\_OUTPUT\_NB):
  
 if (not CCworkOvertime(k)
  
 and not MatchRebootCondition(k)
  
 and VIOM2OutNotDisabled(k)[port]):
  
 VIOM2VitalOut[port] = Offline.GetVIOM2VitalOut(port)
  
 else:
  
 VIOM2VitalOut[port] = False
  
 return 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 514 Configurable outputs for rolling stock

|  |  |  |
| --- | --- | --- |
| Name | Description | InhibitEmergencyBrake |
| 是否禁止输出EB | InhibitParkingBrake | 是否禁止输出PB |
| EnableTrainDoorOpening\_A | 是否授权A侧开门 | EnableTrainDoorOpening\_B |
| 是否授权B侧开门 | HoldDoorsClosed\_A | 是否保持A侧门锁闭 |
| HoldDoorsClosed\_B | 是否保持B侧门锁闭 | EmergencyDetrainDoorLockingEnd1 |
| 是否保持End1端门锁闭 | EmergencyDetrainDoorLockingEnd2 | 是否保持End2端门锁闭 |
| TrainFilteredStopped | 是否ATP判断停车 | TractionAuthorisedSenseEnd1 |
| 是否授权向End1端牵引 | TractionAuthorisedSenseEnd2 | 是否授权向End2端牵引 |

[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，上下CPU模块的周期同步校核字Trace和Dt，安全时钟状态SafeTimerFailed，是否主控CC信息MasterCCcore，以及需要反馈给各自VIOM的时间信息LatestVIOM1LoopHourVIOM[2]和LatestVIOM2LoopHourVIOM[2]。
According to [REF5], the commands ATP sent to VIOM shall attach the current ATPtime, cycle synchronization check word Trace and Dt, safety clock state SafeTimerFailed, MasterCCcore, the LatestVIOM1LoopHourVIOM and 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 | √ | √ | REF BOOLEAN BOOLEAN |
| InhibitParkingBrake | √ | √ | REF BOOLEAN BOOLEAN |
| OutOfCode | √ | √ | REF BOOLEAN BOOLEAN |
| VIOM1VitalOut | √ | √ | REF BOOLEAN BOOLEAN |
| VIOM2VItalOut | √ | √ | REF BOOLEAN 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 as far as possible. After ATP powered up and the communication with LC had fulfilled, and get the current ZC id, it shall communicate with ZC. When the train passes across two ZC, ATP should also send the Location Report to both ZC. The information of Location Report includes train recognition information, ATP status, and train location, so that ZC can calculate the AP scope, properties and EOA of the train.

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATOcontrolTimeValid | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| ATPsetting | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| BeaconLastObtained | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| DateSynchronizationReport | External | REF \_Ref348814411 \h Interface with LC |
| EnableDoorOpening\_A | Internal | REF \_Ref268160839 \h F61-Elaborate Door Opening Authorization |
| EnableDoorOpening\_B | Internal | REF \_Ref268160839 \h F61-Elaborate Door Opening Authorization |
| LocationNotUncertaintyExceed | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| MasterCCcore | Internal | REF \_Ref345055899 \h F62-Control PSD Opening and Closing Order |
| MotionProtectionInhibition | Internal | REF \_Ref345055806 \h F46-Determine Over Energy |
| NewBeaconObtained | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| NonVitalRequest | External | REF \_Ref348357813 \h Interface with CC Non Vital |
| Offline | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| RMRselectedDrivingMode | Internal | REF \_Ref345055856 \h F54-Monitor Rollback Train Speed |
| TrainCoupledType | Internal | REF \_Ref345055566 \h \* MERGEFORMAT F12-Manage Train Status |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFilteredStopped | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainFrontEnd | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainFrontLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainLocalized | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainLocatedOnKnownPath | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| TrainLocation | Internal | REF \_Ref345055736 \h F32-Update Train Location |
| TrainMaxSpeed | Internal | REF \_Ref345055703 \h F27-Compute Train Kinematics |
| TrainRearLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |
| VersionAuthorizedByLC | Internal | REF \_Ref345055594 \h F15-Manage Variants in CBTC Mode |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| DateSynchronisationReportAvailable | √ | √ | REF BOOLEAN BOOLEAN |
| DateSynchronizationReceived | √ | √ | REF BOOLEAN BOOLEAN |
| EnableSendLocReport | √ | √ | REF BOOLEAN BOOLEAN |
| GroundTimeReference | √ | √ | REF ST\_GROUND\_TIME \h ST\_GROUND\_TIME |
| LCsynchronisationLoopHourValid | √ | √ | REF BOOLEAN BOOLEAN |
| LastSynchronisationReportAge | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| NonVitalRouteSetNotNeeded | √ | √ | REF BOOLEAN BOOLEAN |
| SendLocReportOnZCunderTrainHead | √ | √ | REF BOOLEAN BOOLEAN |
| SendLocReportOnZCunderTrainTail | √ | √ | REF BOOLEAN BOOLEAN |
| SignalOverrideSendable | √ | √ | REF BOOLEAN BOOLEAN |
| TimeElapseBetweenTwoLocReport | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| ZCidUnderTrainHead | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| ZCidUnderTrainTail | √ | √ | REF NUMERIC\_32 \h 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. 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, 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 &quot;stretched&quot; to the both ends. The rules of conversion are as following ARDL:

def TrainHeadMinLocation(k):
  
 if (not TrainLocalized(k)):
  
 TrainHeadMinLocation.Block = 0
  
 TrainHeadMinLocation.Abscissa= 0
  
 TrainHeadOrientation = LOCREPORT\_DIRECTION\_UNKNOWN
  
 elif (TrainFrontLocation(k).Min.Ort is UP):
  
 TrainHeadMinLocation.Block = TrainFrontLocation(k).Min.Block(k)
  
 TrainHeadMinLocation.Abscissa = (round.floor(TrainFrontLocation(k).Min.Abscissa(k)
  
 / ABSCISSA\_TO\_HALF\_METER))
  
 TrainHeadOrientation = LOCREPORT\_DIRECTION\_UP
  
 elif (TrainFrontLocation(k).Min.Abscissa(k) + ABSCISSA\_TO\_HALF\_METER
  
 &lt;= TrackMap.Blocks[TrainFrontLocation(k).Min.Block].Length):
  
 TrainHeadMinLocation.Block = TrainFrontLocation(k).Min.Block(k)
  
 TrainHeadMinLocation.Abscissa = round.ceil(TrainFrontLocation(k).Min.Abscissa(k)
  
 / ABSCISSA\_TO\_HALF\_METER)
  
 TrainHeadOrientation = LOCREPORT\_DIRECTION\_DOWN
  
 else:
  
 NextBlock = TrackMap.NextBlock(TrackMap.Blocks[TrainFrontLocation(k).Min.Block], UP)
  
 if TrackMap.ExistThePole(TrainFrontLocation(k).Min.Block, NextBlock.Id):
  
 TrainHeadMinLocation.Block = NextBlock.Id
  
 TrainHeadMinLocation.Abscissa = round.floor(NextBlock.Length
  
 / ABSCISSA\_TO\_HALF\_METER)
  
 TrainHeadOrientation = LOCREPORT\_DIRECTION\_UP
  
 else:
  
 TrainHeadMinLocation.Block = NextBlock.Id
  
 TrainHeadMinLocation.Abscissa = 0
  
 TrainHeadOrientation = LOCREPORT\_DIRECTION\_DOWN
  
 return 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.

def TrainHeadCoupledStatus(k):
  
 if (TrainFrontEnd(k) is END\_1):
  
 return ((TrainCoupledType(k) is TRAIN\_NOT\_COUPLED)
  
 or (TrainCoupledType(k) is TRAIN\_COUPLED\_END2))
  
 else:
  
 return ((TrainCoupledType(k) is TRAIN\_NOT\_COUPLED)
  
 or (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.

def TrainTailCabId(k):
  
 if (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. 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, 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 &quot;stretched&quot; to the both ends. The rules of convertion are as following ARDL:

def TrainTailMinLocation(k):
  
 if (not TrainLocalized(k)):
  
 TrainTailMinLocation.Block = 0
  
 TrainTailMinLocation.Abscissa = 0
  
 TrainTailOrientation = LOCREPORT\_DIRECTION\_UNKOWN
  
 elif (TrainRearLocation(k).Min.Ort is DOWN):
  
 TrainTailMinLocation.Block = TrainRearLocation(k).Min.Block
  
 TrainTailMinLocation.Abscissa = (round.floor(TrainRearLocation(k).Min.Abscissa
  
 / ABSCISSA\_TO\_HALF\_METER))
  
 TrainTailOrientation = LOCREPORT\_DIRECTION\_DOWN
  
 elif (TrainRearLocation(k).Min.Abscissa + ABSCISSA\_TO\_HALF\_METER
  
 &lt;= TrackMap.Block[TrainRearLocation(k).Min.Block].Length):
  
 TrainTailMinLocation.Block = TrainRearLocation(k).Min.Block
  
 TrainTailMinLocation.Abscissa = round.ceil(TrainRearLocation(k).Min.Abscissa
  
 / ABSCISSA\_TO\_HALF\_METER)
  
 TrainTailOrientation = LOCREPORT\_DIRECTION\_UP
  
 else:
  
 NextBlock = TrackMap.NextBlock(TrackMap.Blocks[TrainRearLocation(k).Min.Block], UP)
  
 if (NextBlock is None):
  
 TrainTailMinLocation.Block = TrainFrontLocation(k).Min.Block
  
 TrainTailMinLocation.Abscissa = round.ceil(TrainRearLocation(k).Min.Abscissa
  
 / ABSCISSA\_TO\_HALF\_METER)
  
 TrainTailOrientation = LOCREPORT\_DIRECTION\_UP
  
 elif (TrackMap.ExistThePole(TrainRearLocation(k).Min.Block, NextBlock.Id)):
  
 TrainTailMinLocation.Block = NextBlock.Id
  
 TrainTailMinLocation.Abscissa = round.floor(NextBlock.Length
  
 / ABSCISSA\_TO\_HALF\_METER)
  
 TrainTailOrientation = LOCREPORT\_DIRECTION\_DOWN
  
 else:
  
 TrainTailMinLocation.Block = NextBlock.Id
  
 TrainTailMinLocation.Abscissa = 0
  
 TrainTailOrientation = LOCREPORT\_DIRECTION\_UP
  
 return 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.

def TrainTailCoupledStatus(k):
  
 if (TrainFrontEnd(k) is END\_1):
  
 return ((TrainCoupledType(k) is TRAIN\_NOT\_COUPLED)
  
 or (TrainCoupledType(k) is TRAIN\_COUPLED\_END1))
  
 else:
  
 return ((TrainCoupledType(k) is TRAIN\_NOT\_COUPLED)
  
 or (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. 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 &quot;overestimated&quot;.

def LocationError(k):
  
 if TrainLocalized(k):
  
 return round.ceil((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, ATP can judge whether it is necessary to send RSNN information.

def RouteSetNotNeededSendable(k):
  
 return Offline.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(k):
  
 return (ATOcontrolTimeValid(k)
  
 and 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.

def TrainRouteSetNotNeeded(k):
  
 return (TrainFilteredStopped(k)
  
 and NonVitalRouteSetNotNeeded(k)
  
 and 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.

def TrainCorrectDocking(k):
  
 return (EnableDoorOpening\_A(k) or 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. 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 &quot;overestimated&quot;.

def LocReportSpeed(k):
  
 return round.ceil(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.

def TrainMonitoringMode(k):
  
 if (MotionProtectionInhibition(k)
  
 and RMRselectedDrivingMode(k)):
  
 return RMR
  
 elif (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(k):
  
 return Offline.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.

def SignalsOverride(k):
  
 return (SignalOverrideSendable(k)
  
 and not MotionProtectionInhibition(k)
  
 and (ATOcontrolTimeValid(k)
  
 and 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(k):
  
 return Offline.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(lcId, k):
  
 return Message.Received(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 used to judge whether the LC synchronization message could use or not. When the below conditions fulfilled, DateSynchronisationReportAvailable shall set as True. Otherwise, it set as False.  
If ATP Receives the 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(lcId, k):
  
 return Message.Available(DateSynchronizationReceived(lcId, k),
  
 DateSynchronizationReport(lcId).CcLoopHour,
  
 ATPsetting.LCloophourValidityTime,
  
 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 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(lcId, k):
  
 return Message.LastAge(DateSynchronisationReportAvailable(lcId, k),
  
 DateSynchronizationReport(lcId).CcLoopHour,
  
 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 will check whether the LC message is still valid.

def LCsynchronisationLoopHourValid(lcId, k):
  
 return Message.Valid(DateSynchronizationReport(lcId).CcLoopHour,
  
 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 | REF NUMERIC\_32 \h NUMERIC\_32 | 以轨旁周期号为单位的时间 | Milliseconds |

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 for trackside equipment (ZC /LC). From power-up, 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(LcId, k) != True)
  
 GroundTimeReference[LcId].Time = INVALID\_LC\_DATE
  
 GroundTimeReference[LcId].Milliseconds = 0
  
 elif (DateSynchronisationReportAvailable(LcId, k))
  
 GroundTimeReference[LcId].Time(k)= DateSynchronizationReport.Synchrodate
  
 GroundTimeReference[LcId].Milliseconds = 0
  
 else:
  
 GroundTimeReference[LcId].Time(k)
  
 = GroundTimeReference[LcId].Time(k-1)
  
 + ((GroundTimeReference[LcId].Milliseconds(k-1)+ ATP\_CYCLE\_TIME\_MS)
  
 / SYNCHRODATE\_TIME\_UNIT\_MS)
  
 GroundTimeReference[LcId].Milliseconds(k)
  
 = ((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参与计算，仅发送给ZC使用。对于每个ZC的LocReport，应当发送其所对应的LC的时间。
In ATP shall not use GroundTimeReference during the calculation of the internal functions, but only send the info to 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(LcId, k):
  
 return 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的时间控制：
ATP shall calculate when to send the Location Report as the following ARDL:

def TimeElapseBetweenTwoLocReport(k):
  
 if (Initialization
  
 or (TimeElapseBetweenTwoLocReport(k-1) == 1)):
  
 return round.floor(ATPsetting.LocReportEmissionPeriod)
  
 else:
  
 return 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 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 area id.

def ZCidUnderTrainTail(k):
  
 if (Initialization):
  
 return 0
  
 elif (TrainLocalized(k)):
  
 return TrackMap.ZCId(TrainTailMinLocation(k).Block)
  
 elif (MemorizedLocationAvailable(k)):
  
 if (TrainFrontEnd(k) is END\_1):
  
 return TrackMap.ZCId(MemLocation.Ext2.Block)
  
 else:
  
 return TrackMap.ZCId(MemLocation.Ext1.Block)
  
 elif (NewBeaconObtained(k)):
  
 return TrackMap.ZCId(BeaconLastObtained.Block)
  
 else:
  
 return 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 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 area id.

def ZCidUnderTrainHead(k):
  
 if (Initialization)
  
 return 0
  
 elif (TrainLocalized(k)):
  
 return TrackMap.ZCId(TrainHeadMinLocation(k).Block)
  
 elif (MemorizedLocationAvailable(k)):
  
 if (TrainFrontEnd(k) is END\_1):
  
 return TrackMap.ZCId(MemLocation.Ext1.Block)
  
 else:
  
 return TrackMap.ZCId(MemLocation.Ext2.Block)
  
 elif (NewBeaconObtained(k)):
  
 return TrackMap.ZCId(BeaconLastObtained.Block)
  
 else:
  
 return 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 is valid and not as same as the train head ZC, and the ZC train tail located authorized by the LC, ATP shall send the Location Report to this ZC where train tail located.

def SendLocReportOnZCunderTrainTail(k):
  
 return (((TimeElapseBetweenTwoLocReport(k) == 1)
  
 or (not MasterCCcore(k-1) and MasterCCcore(k)))
  
 and (ZCidUnderTrainTail(k) is not None)
  
 and (ZCidUnderTrainTail(k) != ZCidUnderTrainHead(k))
  
 and EnableSendLocReport(TrackMap.ZC[ZCidUnderTrainTail(k)].LcId, k)
  
 and (VersionAuthorizedByLC(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。
When the time elapsed from last report is equal to the Location Report sending cycle, and the train head located ZC is valid, and the ZC train head located authorized by the LC, ATP shall send the Location Report to this ZC where train head located.

def SendLocReportOnZCunderTrainHead(k):
  
 return (((TimeElapseBetweenTwoLocReport(k) == 1)
  
 or (not MasterCCcore(k-1) and MasterCCcore(k)))
  
 and (ZCidUnderTrainHead(k) is not None)
  
 and EnableSendLocReport(TrackMap.ZC[ZCidUnderTrainHead(k)].LcId, k)
  
 and (VersionAuthorizedByLC(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 | √ | √ | REF BOOLEAN BOOLEAN |
| LocReportSpeed | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| LocationError | √ | √ | REF BOOLEAN BOOLEAN |
| RouteSetNotNeededSendable | √ | √ | REF BOOLEAN BOOLEAN |
| SignalsOverride | √ | √ | REF BOOLEAN BOOLEAN |
| TrainCorrectDocking | √ | √ | REF BOOLEAN BOOLEAN |
| TrainHeadCoupledStatus | √ | √ | REF BOOLEAN BOOLEAN |
| TrainheadMinLocation | √ | √ | REF ST\_LOCATION\_UNIT \h ST\_LOCATION\_UNIT |
| TrainHeadOrientation | √ | √ | REF ENUM\_DOT \h ENUM\_DOT |
| TrainMonitoringMode | √ | √ | REF ENUM\_MONITORING\_MODE \h ENUM\_MONITORING\_MODE |
| TrainRouteSetNotNeeded | √ | √ | REF BOOLEAN BOOLEAN |
| TrainTailCabId | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| TrainTailCoupledStatus | √ | √ | REF BOOLEAN BOOLEAN |
| TrainTailMinLocation | √ | √ | REF ST\_LOCATION\_UNIT \h ST\_LOCATION\_UNIT |
| TrainTailOrientation | √ | √ | REF ENUM\_DOT \h ENUM\_DOT |

## F73-Outputs to CI Radio

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| CCvariantRequest | External | REF \_Ref348813931 \h Interface with CI Radio |
| NonVitalRequest | External | REF \_Ref348357813 \h Interface with CC Non Vital |
| TrackMap | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| TrainFrontLocation | Internal | REF \_Ref345055747 \h F33-Confirm Train Localization |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| CCvariantRequestMsgReceived | √ | √ | REF BOOLEAN 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(cbiId, k):
  
 if (Initialization):
  
 return False
  
 elif (Message.Received(CCvariantRequest, k)):
  
 return True
  
 else:
  
 return 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(cbi, k):
  
 if (Initialization):
  
 return INVALID\_LOOP\_HOUR
  
 elif (CCvariantRequestMsgReceived(Cbi, k)):
  
 return CCvariantRequest(Cbi, k).CbiLoopHour
  
 else:
  
 return 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(CbiId, k):
  
 Orz = TrackMap.ExistZoneLocationIncluded(SGL\_OVERLAP\_RELEASE\_ZONE,
  
 TrainFrontLocation(k).Max):
  
 if (OverlapReleasable(k)
  
 and OtherATP(k).OverlapExpired
  
 and CCvariantRequestMsgReceived(CbiId, k)
  
 and Orz is not None
  
 and Orz.CbiId == CbiId
  
 and CbiId == NonVitalRequest(k).VariantRequestCbiId):
  
 for Index in range(0, MAX\_CC\_VARIANTS\_NB):
  
 if (Orz.RadioBlockModeVariantIndex == Index):
  
 CCvariants[CbiId].Variant[Index] = True
  
 else:
  
 CCvariants[CbiId].Variant[Index] = False
  
 else:
  
 CCvariants[CbiId] = None
  
 return 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 | √ | √ | REF BOOLEAN \h BOOLEAN |
| ReceivedCBIloopHour | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |

## F8-Platform Relative & Assist Functions

Figure 522 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 | External | REF \_Ref348720966 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，生成DataPlugContent，其结构如Table 44所示。
In Initialization, ATP reads the CCdataPlugInfo with VCP coded from CC data plug, and generates DataPlugContent with the structure shown as Table 44。
#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.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.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的部分内容，如Table 47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 47 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.DLUIpBlue和DLUIpRed信息通过双口RAM发送给DLU。
In initialization, ATP needs to send the 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. 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 | √ | √ | REF ST\_DATA\_PLUG \h ST\_DATA\_PLUG |

## F82-Manage Vital Time

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| ATOcontrolTimeValid | Internal | REF \_Ref345055549 \h F11-Acquire Configuration Data |
| DataPlugContent | Internal | REF \_Ref345055964 \h F81-Initialize ATP software |
| NonVitalRequest | External | REF \_Ref348357813 \h Interface with CC Non Vital |
| TOC\_VitalTime | External | REF \_Ref348814578 \h Interface between two CPUs |
| VLE\_GetSafeTime | External | REF \_Ref348814596 \h Interface with VLE Hardware |

### Locals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Safety Critical | Observable | Logical Type |
| CycleBiasNb | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| CycleSynchronized | √ | √ | REF BOOLEAN \h BOOLEAN |
| ImmediateCounter | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| LockedImmediateCounter | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| VLEimpulseNb | √ | × | REF NUMERIC\_32 \h 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接口获取VLE的脉冲数，将其存储在VLEimpulseNb数组中。
VLEimpulseNb[ATP\_INTERRUPT\_NB] array stores the safe clock impulse number for every interrupt triggered. ATP shall obtain the impulse number through 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.VLECpuId == CPU1)
  
 Dt = CPU1\_DT\_INIT
  
 Trace = CPU1\_TRACE\_0 ^ Dt
  
 m = InversePDoperation(CPU1\_TRACE\_0, CPU1\_TRACE\_N)
  
 else:
  
 Dt = CPU2\_DT\_INIT
  
 Trace = CPU2\_TRACE\_0 ^ Dt
  
 m = InversePDoperation(CPU2\_TRACE\_0, CPU2\_TRACE\_N)

否则，如果ImmediateNb = 0，而VitalTime与上个中断相比仍然未发生变化，则
将CycleBiasNb加1。
如果CycleBiasNb &gt; 1，设置CycleSynchronized为False；
否则，如果ImmediateNb = 0，而VitalTime与上个中断相比发生了变化，则
令ImmediateNb = 1；
令Trace = PDoperationDt(Trace，Bi[ImmediateNb], Dt)
设置CycleBiasNb = 0，而且CycleSynchronized为True.
使用LockedImmediateCounter锁存此时的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, Bi[ImmediateNb], Dt)；
如果ImmediateNb &gt; (ATP\_INTERRUPT\_NB-1)，则设置ImmediateNb = 0；并令Trace = PDoperation(Trace，m)，令Dt = PDoperation(Dt, 0)
When the fixed-time interrupt triggered, ATP shall update the ImmediateNb，CycleSynchronized, Trace, Dt and CycleBiasNb.
If it is the first interrupt after powered up, ATP shall set ImmediateNb as zero，and initialize the Trace, Dt (the dynamic time of VCP), and the middle variables m based on CPU.

if (DataPlugContent.VLECpuId == CPU1)
  
 Dt = CPU1\_DT\_INIT
  
 Trace = CPU1\_TRACE\_0 ^ Dt
  
 m = InversePDoperation(CPU1\_TRACE\_0, CPU1\_TRACE\_N)
  
 else:
  
 Dt = CPU2\_DT\_INIT
  
 Trace = CPU2\_TRACE\_0 ^ Dt
  
 m = InversePDoperation(CPU2\_TRACE\_0, CPU2\_TRACE\_N)

Or else:, If the ImmediateNb is zero, but the VitalTime has not changed comparing to the previous interrupt, then:

CycleBiasNb = CycleBiasNb + 1
  
 if (CycleBiasNb &gt; 1)
  
 CycleSynchronized = False

Or else, If ImmediateNb is zero, and the VitalTime has changed comparing to the previous interrupt, then:

ImmediateNb = 1
  
 Trace = PDoperationDt(Traceï¼ŒBi[ImmediateNb], Dt)
  
 CycleBiasNb = 0
  
 CycleSynchronized = True
  
 LockedImmediateCounter = ImmediateCounter

Otherwise, set:

ImmediateNb = ImmediateNb + 1
  
 Trace = PDoperationDt(Trace, Bi[ImmediateNb], Dt)

and if the ImmediateNb is greater than (ATP\_INTERRUPT\_NB-1), then:

ImmediateNb = 0
  
 Trace = PDoperation(Traceï¼Œm)
  
 Dt = PDoperation(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；PDoperationDt() stands for the PD calculation with 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 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 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 as False
Otherwise, it will set 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和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(k)和Dt(k)作为校核字送给VIOM进行校验。
在本周期最后，设置

VitalTime = VitalTime(k-1) + 1

否则，继续等待。
The 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 and 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 to the other CPU,
and send Trace and Dt to VIOM to check,
and at the end of this cycle, set

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 = True,
and TOC\_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 is True,
and the TOC\_VitalTime get from the other CPU is equal to 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]
Figure 523 Synchronize cycle sequence between two CPUs
NOTES:
如Figure 523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 523, 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 &quot;waiting&quot; 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 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 = VLE\_RTCtime
  
 elif ((ATOcontrolTimeValid(k) == True)
  
 and (NonVitalRequest.NtpTime != None)
  
 and (|NonVitalRequest.NtpTime - RTCtime(k-1)| &gt; MAX\_NTP\_TIME\_ERROR))
  
 RTCtime = NonVitalRequest.NtpTime
  
 else:
  
 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 | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| ImmediateNb | √ | × | REF NUMERIC\_32 \h NUMERIC\_32 |
| RTCtime | × | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| SafeTimerFailed | √ | √ | REF BOOLEAN \h BOOLEAN |
| Trace | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |
| VitalTime | √ | √ | REF NUMERIC\_32 \h NUMERIC\_32 |

## F83-Manage Synchronization between two CPUs

### Inputs

|  |  |  |
| --- | --- | --- |
| Name | Nature | Provided by |
| CBIvariantRequest | External | REF \_Ref348814663 \h Interface with CI Radio |
| CCsynchroReport | External | REF \_Ref348814755 \h Interface with Distant ATP |
| CCvariantReport | External | REF \_Ref348814777 \h Interface with CI Radio |
| CIsetting | External | REF \_Ref348815425 \h Interface with PSD |
| LocReport | External | REF \_Ref348853347 \h Interface with ZC |
| TOC\_CBIvariantRequest | External | REF \_Ref348814578 \h Interface between two CPUs |
| TOC\_CCsyncReport | External | REF \_Ref348814578 \h Interface between two CPUs |
| TOC\_CCvariantReport | External | REF \_Ref348814578 \h Interface between two CPUs |
| TOC\_CIsetting | External | REF \_Ref348814578 \h Interface between two CPUs |
| TOC\_LocReport | External | REF \_Ref348814578 \h Interface between two CPUs |
| TOC\_VersionReport | External | REF \_Ref348814578 \h Interface between two CPUs |
| TOC\_VIOM2VitalOut | External | REF \_Ref348814578 \h Interface between two CPUs |
| VersionFromCCreport | External | REF \_Ref348853436 \h Interface with LC |
| VIOM1VitalOut | External | REF \_Ref345055931 \h F71-Outputs to Rolling-stock |
| VIOM2VitalOut | External | REF \_Ref345055931 \h F71-Outputs to Rolling-stock |

### Processing

如Figure 524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 524, the two set of ATP software in the same VLE-2 board will calculate the output results to VIOM, redundant ATP, 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 524 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和VIOM2VitalOut发送给另一个CPU，并获取来自另一个CPU的运算结果TOC\_VIOM1VitalOut和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 and VIOM2VitalOut to the other CPU through dual-ports RAM, and receive the TOC\_VIOM1VitalOut and 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和TOC\_VIOM1VitalOut按照ST\_VIOM\_OUT结构组合成IdenticalVIOM1Out，将VIOM2VitalOut和TOC\_VIOM2VitalOut按照ST\_VIOM\_OUT结构组合成IdenticalVIOM2Out。
Based on ST\_VIOM\_OUT structure, ATP combines VIOM1VitalOut and TOC\_VIOM1VitalOut as IdenticalVIOM1Out, while based on the same structure ST\_VIOM\_OUT, ATP combines VIOM2VitalOut and TOC\_VIOM2VitalOut as 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后，需将其通过双口RAM发送给对方；并接收来自对方的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, the location report after merging two CPU’s results. When the two CPUs complete LocReport calculation, need to send each other through the dual-port RAM; and receive the TOC\_LocReport from the other. ATP shall combine the two reports as an 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.
[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后，需将其通过双口RAM发送给对方；并接收来自对方的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, the version report after merging two CPU’s results. When the two CPUs complete VersionFromCCreport calculation, need to send each other through the dual-port RAM; and receive the TOC\_VersionReport from the other. ATP shall combine the two reports as an 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后，需将其通过双口RAM发送给对方；并接收来自对方的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, the redundant ATP report after merging two CPU’s results. When the two CPUs complete CCsynchroReport calculation, need to send each other through the dual-port RAM; and receive the TOC\_CCsyncReport from the other. ATP shall combine the two reports as an 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后，需将其通过双口RAM发送给对方；并接收来自对方的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, the PSD control message after merging two CPU’s results. When the two CPUs complete CIsetting calculation, need to send each other through the dual-port RAM; and receive the TOC\_CIsetting from the other. ATP shall combine the two reports as an 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后，需将其通过双口RAM发送给对方；并接收来自对方的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, the CBI variants request after merging two CPU’s results. When the two CPUs complete CBIvariantRequestcalculation, need to send each other through the dual-port RAM; and receive the TOC\_CCvariantReport from the other. ATP shall combine the two reports as an 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后，需将其通过双口RAM发送给对方；并接收来自对方的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, the CC overlap releasable report after merging two CPU’s results. When the two CPUs complete CCvariantReportcalculation, need to send each other through the dual-port RAM; and receive the TOC\_CCvariantReport from the other. ATP shall combine the two reports as an 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 | √ | × | REF ST\_CBI\_VARIANT\_REQUEST \h \* MERGEFORMAT ST\_CBI\_VARIANT\_REQUEST |
| IdenticalCCsyncReport | √ | × | REF ST\_SYNCHRO\_REPORT \h \* MERGEFORMAT ST\_SYNCHRO\_REPORT |
| IdenticalCCvariantReport | √ | × | REF ST\_CC\_VARIANT\_REPORT \h \* MERGEFORMAT ST\_CC\_VARIANT\_REPORT |
| IdenticalCIsetting | √ | × | REF ST\_CI\_SETTING \h \* MERGEFORMAT ST\_CI\_SETTING |
| IdenticalLocReport | √ | × | REF ST\_LOC\_REPORT \h \* MERGEFORMAT ST\_LOC\_REPORT |
| IdenticalVIOM1Out | √ | × | REF ST\_VIOM\_OUT \h \* MERGEFORMAT ST\_VIOM\_OUT |
| IdenticalVIOM2Out | √ | × | REF ST\_VIOM\_OUT \h \* MERGEFORMAT ST\_VIOM\_OUT |
| IdenticalVersionReport | √ | × | REF ST\_VERSION\_REPORT \h \* MERGEFORMAT ST\_VERSION\_REPORT REF ST\_VERSION\_REPORT \h \* MERGEFORMAT ST\_VERSION\_REPORT |