SIGSCRIPT Functions

Below is an overview of the functions available in OR for use in signal scripts.

Original MSTS Functions

The following are basic MSTS functions.
BLOCK_STATE
ROUTE_SET
NEXT_SIG_LR
NEXT_SIG_MR
THIS_SIG_LR
THIS_SIG_MR
OPP_SIG_LR
OPP_SIG_MR
DIST_MULTI_SIG_MR
SIG_FEATURE
DEF_DRAW_STATE

Extended MSTS Functions

The following are extensions of basic MSTS functions.

NEXT_NSIG_LR(SIGFN_TYPE, N)

Extension of NEXT_SIG_LR. Returns state of Nth signal of type SIGFN_TYPE.
Note that state SIGASP_STOP is returned if any intermediate signal of type SIGFN_TYPE is set to that state.

DIST_MULTI_SIG_MR_OF_LR(SIGFN_TYPE, SIGFN_ENDTYPE)

Extension of DIST_MULTI_SIG_MR.
The original DIST_MULTI_SIG_MR excluded any heads for which the link (route_set) was not valid. However, when signals are routed through route-definition signals rather than through links, this exclusion fails and therefor the function does not return the correct state.
This extended function checks all required heads on each signal, and uses the least restricted aspect on this signal as state for this signal. It returns the most restrictive state of the states determined in this manner for each intermediate signal until a signal of type SIGFN_ENDTYPE is found.

SIGNAL IDENT Functions

When a function is called which requires information from a next signal, a search is performed along the train's route to locate the required signal. If multiple information is required from that signal, and therefor multiple functions are called requiring that next signal, such a search is performed for each function call.

This process can be made much more efficient by using the signal ident of the required signal. Each signal in a route has a unique ident. A set of functions is available to obtain the signal ident of the required signal. Also available are functions which are equivalent to normal signal functions, but use the signal ident and do not perform a search for the required signal.
Obviously, using these functions it must be checked that the retrieved signal ident is valid (i.e. a valid signal is found), and the integrity of the variable holding this ident must be ensured (the value must never be altered).

The following functions are available to obtain the required signal ident.
The functions return the signal ident for the signal as found. If no valid signal is found, the value of -1 is returned.

NEXT_SIG_ID(SIGFN_TYPE)
Returns Signal Ident of next signal of type SIGFN_TYPE.

NEXT_NSIG_ID(SIGFN_TYPE, N)
Returns Signal Ident of Nth signal of type SIGFN_TYPE.

OPP_SIG_ID(SIGFN_TYPE)
Returns Signal Ident next signal of type SIGFN_TYPE in opposite direction.

The following functions are equivalent to basic functions but use Signal Ident to identify the required signal.

ID_SIG_ENABLED(SigID)
Returns 1 if the identified signal is actively enabled (i.e. a train has cleared a route leading to that signal)

ID_SIG_LR(SigId, SIGFN_TYPE)
Returns the least restricted aspect of the heads with type SIGFN_TYPE of the identified signal.

Note there are other functions which also use the signal ident as detailed below.

Signal SubObject functions

In the original MSTS signal definition, a number of specific Signal SubObjects could be used as flags (USER_1 … USER_4). Other items (NUMBER_PLATE and GRADIENT) could also be used as flag but were linked to physical items on the signal. The number of flags available in this way was very restricted.
In OR, an additional functions has been created which can check for any Signal SubObject if this SubObject is included for this particular signal or not. This function can be used on any type of Signal SubObject. By setting SubObjects of type ‘DECOR’, additional flags can be defined for any type of signal. SubObjects defined in this manner need not be physically defined in the shape file. The information is available at signal level, so all heads on a signal can use this information.
The function uses the SubObject number to identify the required SubObject, the name of the SubObject is irrelevant. The maximum of total SubObjects for any shape is 32 (no. 0 … 31), this includes the actual signal heads.

HASHEAD(N)
Returns ‘true’ (1) if SubObject with number N is available on this signal.

Approach Control Functions
Approach Control is a method used in some signalling systems which holds a signal at danger until the approaching train is at a specific distance from the signal, or has reduced its speed to below a certain limit. This functionality is used in situations where a significant reduction in speed is required for the approaching train, and keeping the signal at danger ensures the train has indeed reduced its speed to near or below the required limit.
The following set of Approach Control Functions is available in OR.
The required distance and speed can be set as constants (dimensions are m and m/s, these dimensions are fixed and do not depend on any route setting).
It is also possible to define the required distance or speed in the signal type definition in sigcfg.dat. The values defined in this way can be retrieved using the pre-defined variables
Approach_Control_Req_Position and ***Approach_Control_Req_Speed*** .

APPROACH_CONTROL_POSITION(APPROACH_CONTROL_POSITION)
The signal will be held at danger until the train has reached the distance ahead of the signal as set.
The signal will also be held at danger if it is not the first signal ahead of the train, even if the train is within the required distance.

APPROACH_CONTROL_POSITION_FORCED(APPROACH_CONTROL_POSITION)
The signal will be held at danger until the train has reached the distance ahead of the signal as set. The signal will also clear even if it is not the first signal ahead of the train.

APPROACH_CONTROL_SPEED(APPROACH_CONTROL_POSITION, APPROACH_CONTROL_SPEED)
The signal will be held at danger until the train has reached the distance ahead of the signal as set, and the speed has been reduced to below the required limit.
The speed limit may be set to 0 in which case the train has to come to a stand in front of the signal before the signal will be cleared.
The signal will also be held at danger if it is not the first signal ahead of the train, even if the train is within the required distance.

APPROACH_CONTROL_NEXT_STOP(APPROACH_CONTROL_POSITION, APPROACH_CONTROL_SPEED)
Sometimes, a signal may have approach control but the signal may be held at danger if the next signal is not cleared. Normally, if a signal is held for approach control, it will not propagate the signal request, meaning that the next signal will never clear. This could lead to a signal lockup, with the first signal held for approach control and therefor the next signal cannot clear.
This function is specifically intended for that situation. It will allow propagation of the clear request even if the signal is held at danger for approach control, thus allowing the next signal to clear.
The working of this function is simular to APPROACH_CONTROL_SPEED.

APPROACH_CONTROL_LOCK_CLAIM()
If a signal ahead of a train is held at danger, the train may claim sections beyond that signal in order to ensure a clear path from that signal as soon as possible. If this function is called in a script sequence

which also sets an active approach control, no claims will be made while the signal is held for approach control.

CallOn Functions

CallOn functions allow trains to proceed unto a track section already occupied by another train.
CallOn functions should not be confused with ‘permissive’ signals as often used in North American signal systems.
A ‘permissive’ signal will always allow a train to proceed on occupied track, following a previous train.
Such signals are generally only used in situations where a signal covers a ‘free line’ section only, i.e. a section of track without switches or crossings etc.
The CallOn facility, on the other hand, will only allow the train to proceed in certain specific situations and is primarily used in station and yard areas.
The CallOn functions are specifically intended for use is timetable mode, and are linked directly to a number of timetable commands. Trains will be allowed to proceed based on these commands.

CallOn functions in timetable mode.
The following conditions will allow a train to proceed.

- The route beyond the signal leads into a platform, and the **$callon** parameter is set for the related station stop.
- The train has an **$attach**, **$pickup** or **$transfer** command set for a station stop or in the *#dispose* field, and the train in the section beyond the signal is static or is the train as referenced in the command (as applicable).
  If the command is set for a station stop, the route beyond the signal must lead into a platform allocated to that station.
  If the command is set in the #dispose field, there are no further conditions.
- The train action is part of a **$stable** command in the *#dispose* field.
- The route beyond the signal is a Pool Storage path, and the train is booked to be stored in that pool.

CallOn may also be allowed if the route does not lead into a platform depending on the function call.

CallOn functions in activity mode.
CallOn is never allowed if the route beyond the signal leads into a platform.
CallOn may be allowed in other locations depending on the actual function call.

Available functions :
TRAINHASCALLON()
TRAINHASCALLON_RESTRICTED()
These functions are similar, except that TRAINHASCALLON will always allow CallOn if the route does not lead into a platform, and therefor acts like a ‘permissive’ signal in that situation.
The function `TRAINHASCALLON_RESTRICTED will only allow CallOn when one of the criteria is met as detailed above.`

SignalNumClearAhead Functions

The SignalNumClearAhead (SNCA) value sets the number of signals ahead which a signal will need to clear in order to be able to show the required least restrictive aspect.

The value is set as a constant for each specific signal type in the sigcfg.dat file.
However, it may be that certain signal options require that value to be changed.
For instance, a signal type which optionally can display an advance approach aspect, needs a higher value for SNCA in case this advance approach is required. This may even depend on the route as set from that signal. In OR, functions are available to adjust the value of SNCA as required, which prevents the need to always set the possible highest value which could lead to a signal to clear a route too far ahead.

Note that these functions always use the default value of SNCA as defined in sigscr.dat as starting value.
Repeated calls of these functions will not lead to invalid or absurd values for SNCA.

INCREASE_SIGNALNUMCLEARAHEAD(n)
Increase the value of SNCA by *n*, starting from the default value.

DECREASE_SIGNALNUMCLEARAHEAD(n)
Decrease the value of SNCA by *n*, starting from the default value.

SET_SIGNALNUMCLEARAHEAD(n)
Set the value of SNCA to *n*.

RESET_SIGNALNUMCLEARAHEAD()
Reset the value of SNCA to the default value.

Local signal variables.

Originally, the only means of interfacing between signals, or between signal heads within a signal, is through the signal aspect states. This sets a severe restriction of the amount of information that can be passed between signals or signal heads.
In OR, local signal variables have been introduced. These variables are specific for a signal. The variables are persistent, that is they do retain their value from one update to the next. Because the variables are assigned per signal, they are available to all signalheads which are part of that signal. The variables can also be accessed by other signals.
Each signalhead which is part of a signal can access the variables for both reading and writing.
Each signalhead from other signals can access the variables for reading only.
Each variable is identified by an integer number. The variables can contain integer values only.

STORE_LVAR(IDENT, VALUE)
Sets the variable as identified by ***IDENT*** to ***VALUE***. The function has no return value.

THIS_SIG_LVAR(IDENT)
Returns the value of the variable identified by ***IDENT*** of this signal.

NEXT_SIG_LVAR(SIGFN_TYPE, IDENT)
Returns the value of the variable identified by ***IDENT*** of the first signal ahead of type ***SIGFN_TYPE***.
If no such signal is found, the function returns value 0.

ID_SIG_LVAR(SIGID, IDENT)
Returns the value of the variable identified by ***IDENT*** of the signal identified by the signal ident ***SIGID***.

Functions for Normal Head Subtype

Although there can be different types of signal, and OR allows the definition and additional of any number of type, only signals of type NORMAL will affect the trains.
Certain signal systems, however, have different types of signals, e.g. main and shunt signals, which require different behaviour or different response. In order to be able to distinguish between such signals, OR has introduced a Subtype which can be set for a NORMAL signal.
The subtype can be defined for any signal in the sigcfg.dat file, in the same way as the signal type.
A number of functions is available to query a signal to identify its subtype.

THIS_SIG_HASNORMALSUBTYPE(SIGSUBTYPE)
Returns value 1 (true) if this signal has any head of type NORMAL with the required subtype.

NEXT_SIG_HASNORMALSUBTYPE(SIGSUBTYPE)
Returns value 1 (true) if next signal with any head of type NORMAL has any head with the required subtype.

ID_SIG_HASNORMALSUBTYPE(SIGIDENT, SIGSUBTYPE)
Returns value 1 (true) if signal identified by SIFIDENT has any head of type NORMAL with the required subtype.

Functions to verify full or partial route clearing

As mentioned, some signal systems differentiate between main and shunt signals (e.g. in Germany, UK). This may affect the clearing of a signal in locations where both such types occur on the same route.
If a train requires the full route from a main signal to the next main signal, in locations where there are shunt signals inbetween, the first main signal may not clear until the full route to the next main signal is available, and will then clear to a main aspect. If, however, the train only requires a partial route (e.g. for shunting), the signal may clear as soon as (part of) this route is available, and will generally then clear only to a restricted or auxiliary aspect (shunt aspect).
The original MSTS signal functions could not support such a situation, as the signal would always clear as soon as the first part of the route became available, because it was not possible to distinguish between the different types of signal.
Due to the introduction of the Normal Subtype as detailed above, such a setup is not possible. A number of functions have been introduced to support this.
Use of these functions is, however, fairly complicated, and only a brief description of these functions is provided in this document.

TRAIN_REQUIRES_NEXT_SIGNAL(SIGIDENT, REQPOSITION)
Returns value 1 (true) if train requires the full route to the signal as identified by SIGIDENT.
If REQPOSITION is set to 0, the route is checked up to and including the last section ahead of the relevant signal.
If REQPOSITION is set to 1, the route is checked up to and including the first section immediately behind the relevant signal.

FIND_REQ_NORMAL_SIGNAL(SIGSUBTYPE)
Returns the Signal Ident of the first NORMAL signal which has a head with the required SIGSUBTYPE, or -1 if such a signal cannot be found.

ROUTE_CLEARED_TO_SIGNAL(SIGIDENT)
Returns value 1 (true) if the route as required is clear and available.

ROUTE_CLEARED_TO_SIGNAL_CALLON(SIGIDENT)
As ROUTE_CLEARED_TO_SIGNAL, but will also return value 1 (true) if the route is available because the train is allowed to call-on.

Miscellaneous functions

A number of miscellaneous functions which are not part of any of the groups detailed above.

ALLOW_CLEAR_TO_PARTIAL_ROUTE(SETTING)
If the route of a train passing a signal stops short of the next signal (no further NORMAL signal is found on that route), the relevant signal will only clear if the train is approaching that signal, i.e. it is the first signal in the train's path.
This setting can be overruled by this function.
If SETTING is set to 1, the signal will clear if required and the route is available, even if no further NORMAL signal is found.
If SETTING is set to 0, the normal working is restored.

THIS_SIG_NOUPDATE()
After the signal has been processed once, it will not be updated anymore. This is useful for fixed signals, e.g. at end of track like bufferstop lights, but also for fixed signals like route control or route information signals. Calling this function in the script for such signals excludes these signals from the normal updates which will save processing time. Note that the signals are always processed once, so the script will be executed once to set the signal to the required fixed state.

SWITCHSTAND(ASPECT_STATE_0, ASPECT_STATE _1)
Special functions for signals used as switchstand. A direct link is set between the switch and the signal, such that the signal is immediately updated as and when the state of the switch is changed.
The signal will be set to ASPECT_STATE_0 when the switch is set to route 0, and to ASPET_STATE_1 when the switch is set to route 1. Linking the signal to the switch routes is not necessary.
Using this function for switchstands eliminates the delay which normally can occur between the change of the switch state and the state of the signal, due to the independent processing of the signal.
Note that the signal can be excluded from the normal update process as it will be updated through the direct link with the switch.

OR Specific additions to SIGCFG files

Detailed below are OR specific additions which can be set in the SIGCFG file to set specific characteristics or enhance the functionality of the signal types.

General definitions

The following are general definitions which must be set before the definitions of the signal types, immediately following the lighttextures and lightstab definitions.

ORTSSignalFunctions

Additional signal types can be defined in OR, over and above the standard MSTS signal types.
The additional types must be predefined in the sigcfg.dat file using the ORTSSignalFunctions definition.
The defined ORTS signal types can be set in the signal type definition and used in signal script functions in the same way as the default MSTS types.
Note that SPEED is a fixed signal type which is available in OR without explicit definition (see below for details on SPEED type signals). Also note that any type definition starting with "OR_" is not valid, these names are reserved for future default types in OR.

Syntax :

```
ORTSSignalFunctions ( n
        ORTSSignalFunctionType ( "signaltype" )
        ….
)
```

The value ***n*** indicates the total number of definitions.
The value ***signaltype*** is the name of the additional type.

ORTSNormalSubtypes

As detailed above, subtypes can be defined for NORMAL type signals which allows to distinguish different use of NORMAL type signals.
The Normal Subtype must be predefined in the sigcfg.dat file using the ORTSNormalSubtypes definition.
The Subtype can be set in the type definition for NORMAL type signals using the ORTSNormalSubtype statement, see below.
The subtype can be used in specific signal script functions as detailed above.

Syntax :

```
ORTSNormalSubtypes ( n
        ORTSNormalSubtype ( " subtype " )
        ….
)
```

The value ***n*** indicates the total number of definitions.
The value ***subtype*** is the name of the subtype.

Signal Type definitions

The following section details OR specific additions to the signal type definition.

Glow settings

Signal Glow is a feature in OR to improve the visibility of signals at larger distances.
The required glow setting can de set per signal type in the signal definition.
The value is a real number, and sets the intensity of the glow. Value 0.0 defines that there is no glow effect.
Default program values for glow are :
Day value = 3.0;
Night value = 5.0;

Notes :

- For signal types which have "Semaphore" flag set, the Day value = 0.0.
- For signals of type INFO and SHUNTING, both Day and Night value are set to 0.0 (no glow).

Syntax :

ORTSDayGlow (***d***)
ORTSNightGlow (***n***)

The values ***d*** and ***n*** are the day and night glow values, as real numbers.

Light switch

There were many signalling systems where semaphore signals did not show lights during daytime. This effect can be simulated using the ORTSDayLight setting.

Syntax :

ORTSDayLight(***l***)

The value ***l*** is a logical value, if set to ***false***, the signal will not show lights during daytime.

Script Function

Normally, each signal type must have a linked signal script, with the same name as defined for the signal type. However, often there are a series of signal types which may differ in definition, e.g. due to differences in the position of the lights, but which have the same logic scripts.
In OR, a signal type can have a definition which references a particular script which this signal type must use. Different signal types which have the same logic can therefor all use the same script.
This script may be defined using the name of one of these signal types, or it may have a generic name not linked to any existing signal type.

Syntax :

ORTSScript(***name***)

The value ***name*** is the name of the signal script as defined in the sigscr.dat file.

Normal Subtype

As detailed above, a signal type of type NORMAL may have an additional subtype definition.

Syntax :

ORTSNormalSubtype(***subtype***)

The value ***subtype*** is the subtype name and must match one of the names defined in *ORTSNormalSubtypes*.

Approach Control Settings

The required values for approach control functions for a particular signal type can be defined in the signal type definition. These values can be referenced in the signal script as defined for the approach control functions.

Syntax :

```
ApproachControlSettings (
        PositionDefinition  ( position )
        SpeedDefinition  ( speed )
        )
```

Possible position definitions :

    Positionkm
    Positionmiles
    Positionm
    Positionyd

Possible speed definitions :

    Speedkph
    Speedmph

The value ***position*** is the required position value in dimension as set by the relevant parameter.
The value ***speed*** is the required speed value in dimension as set by the relevant parameter.

Inclusion of speed definition is optional and need not be set if only approach control position functions are used.

Signal aspect parameters

The following parameters can be included in signal aspect definitions.

or_speedreset

Can be used in combination with a speed setting.
Its function, combined with the speed setting, is as follows.
In **activity** mode :

- If set, the speed as set applies until overruled by a speedpost or next signal setting a higher speed value;
- If not set, speed as set applies until the next signal and will not be overruled by a speedpost.

In **timetable** mode :

- Speed as set always applies until overruled by a speedpost or next signal setting a higher speed value, this flag has no effect in timetable mode.

or_nospeedreduction

For signal aspects “STOP_AND_PROCEED” and “RESTRICTING”, trains will reduce speed to a low value on approach of the signal.
If this flag is set, trains are allowed to pass the signal at normal linespeed.

SPEED Signal

A new standard signaltype, “SPEED”, has been added to OR.
Signals defined as type “SPEED” are processed as speedposts and not as signals.
The required speed limit can be set using the speed setting of the signal aspect definition.

The advantages of using “SPEED” signals over speedposts are :

- “SPEED” signals can be scripted, and can therefor be conditional, e.g. a speed restrictions in only set on approach to a junction if a restricted route is set through that junction.
- “SPEED” signals can set a state according to their setting, and this state can be seen by a preceeding signal. This can be used to set up variable speed warning signs.

A “SPEED” signalhead can be part of a signal which also contains other heads, but for clarity of operation this is not advisable.