Research & Vehicle Technology
"Infotainment Systems Product Development"

Feature – Button Strategy

LIN ICP Infotainment Subsystem Part Specific Specification (SPSS)

Version 1.2
UNCONTROLLED COPY IF PRINTED

Version Date: June 30, 2020

FORD CONFIDENTIAL

Revision History

Date	Version	Notes	
December 8, 2014	1.0	Initial Release	
May 18, 2015	1.1		
	BUTTON-SR-REQ-096736/C-LongEvent		<jmyslin2 / hzubert> changed long press from 2.0 to 1.5 seconds
June 30, 2020	1.2		
	STR-174872/B-Button Interface Requirements - LIN		<hzubert> added description for rolling counters, indicators, new illumination strategy and new error codes for new generartion ICPs
	BUTTONv2-IIR-REQ-096644/D-LIN BCP Button Press - Button Interface Requirements		<hzubert> added hint for interfaces having less button slots; corrected 0x255 to 0xFF
	BUTTON-IIR-REQ-153852/B-LIN - ApplicationInformation2		<hzubert> update
	BUTTON-IIR-REQ-372239/A-LIN - ApplicationInformation0_NewGeneration		<hzubert> initial release
	BUTTON-IIR-REQ-372241/A-LIN - ApplicationInformation1_NewGeneration		<hzubert> initial release
	BUTTON-IIR-REQ-372242/A-LIN - ApplicationInformation2_NewGeneration		<hzubert> initial release
	BUTTON-IIR-REQ-372243/A-LIN - ApplicationInformation3_NewGeneration		<hzubert> initial release
	BUTTON-IIR-REQ-107294/C-LIN - LINStatus		<hzubert> revised description to be more precise
	BUTTON-IIR-REQ-366705/A-LIN - ICP_RC		<hzubert> initial release
	BUTTON-IIR-REQ-366707/A-LIN - DSPL_RC		<hzubert> initial release
	BUTTON-IIR-REQ-366710/A-LIN - DSPLIlluIndPTS		<hzubert> initial release
	BUTTON-IIR-REQ-366711/A-LIN - DSPLIlluIndX		<hzubert> initial release
	BUTTON-IIR-REQ-366718/A-LIN - DSPLIlluBtnPTS		<hzubert> initial release
	BUTTON-IIR-REQ-366719/A-LIN - DSPLIlluBtnX		<hzubert> initial release
	BUTTON-IIR-REQ-366725/A-LIN - DSPLIlluVolKnob		<hzubert> initial release
	BUTTON-IIR-REQ-366726/A-LIN - DSPLIlluBtnChrome		<hzubert> initial release
	BUTTON-IIR-REQ-366729/A-LIN - DSPLIlluPWMIndicatorTargetPTS		<hzubert> initial release
	BUTTON-IIR-REQ-366730/A-LIN - DSPLIlluPWMBacklightTargetPTS		<hzubert> initial release
	BUTTON-IIR-REQ-366731/A-LIN - DSPLIlluPWMBacklightTarget		<hzubert> initial release
	BUTTON-IIR-REQ-366732/A-LIN - DSPLIlluPWMIndicatorTarget		<hzubert> initial release
	BUTTON-IIR-REQ-366733/A-LIN - DSPLIlluPWMTimerUp		<hzubert> initial release
	BUTTON-IIR-REQ-366736/A-LIN - DSPLIlluPWMTimerDown		<hzubert> initial release
	BUTTON-IIR-REQ-366738/A-LIN - DSPLIlluDimmingCurveType		<hzubert> initial release
	BUTTON-IIR-REQ-367005/A-LIN - IlluIndAllocX		<hzubert> initial release
	BUTTON-IIR-REQ-367006/A-LIN - IlluBtnAllocX		<hzubert> initial release
	BUTTON-IIR-REQ-367267/A-LIN - ICPIlluSmoothDimmSupp		<hzubert> initial release
	BUTTON-IIR-REQ-366723/A-LIN - ICP_CRC8		<hzubert> initial release
	BUTTON-IIR-REQ-366724/A-LIN - DSPL_CRC8		<hzubert> initial release
	STR-193581/B-General Requirements		<hzubert> added description for rolling counter
	BUTTON-SR-REQ-107308/B-LIN Scheduler turnaround frequency for standard interface		<hzubert> renamed to "LIN Scheduler turnaround frequency for standard interface"; no content change
	BUTTON-SR-REQ-116454/B-LIN Scheduler turnaround default values for standard interface		<hzubert> renamed to "LIN Scheduler turnaround default values for standard interface"; no content change
	BUTTON-SR-REQ-366885/A-Rolling Counter behavior		<hzubert> initial release
	BUTTON-SR-REQ-366886/A-Rolling Counter default value if CRC is supported		<hzubert> initial release
	BUTTON-SR-REQ-366706/A-Rolling Counter default value if CRC not supported		<hzubert> initial release
	BUTTON-SR-REQ-366887/A-Indicator position allocation		<hzubert> initial release
	BUTTON-SR-REQ-366888/A-Button position allocation		<hzubert> initial release
	BUTTON-SR-REQ-383144/A-Illumination configuration		<hzubert> initial release
	BUTTON-SR-REQ-383145/A-Setup illumination allocation		<hzubert> initial release

	BUTTON-SR-REQ-366708/A-CRC8 algorithm if CRC is supported	<hzubert> initial release
	BUTTON-SR-REQ-366709/A-CRC8 notation	<hzubert> initial release
	BUTTON-SR-REQ-366712/A-CRC8 initialization	<hzubert> initial release
	BUTTON-SR-REQ-366713/A-CRC8 computation	<hzubert> initial release
	BUTTON-SR-REQ-366714/A-CRC8 calculation	<hzubert> initial release
	BUTTON-SR-REQ-366715/A-CRC8 unused bits	<hzubert> initial release
	BUTTON-SR-REQ-366716/A-CRC8 unused bytes	<hzubert> initial release
	BUTTON-SR-REQ-366720/A-CRC8 data stream	<hzubert> initial release
	BUTTON-SR-REQ-366721/A-CRC8 algorithm if CRC is not supported	<hzubert> initial release
	BUTTON-SR-REQ-366722/A-decision of interface variant	<hzubert> initial release
	BUTTON-SR-REQ-372268/A-standard interface	<hzubert> initial release
	BUTTON-SR-REQ-372270/A-new generation interface	<hzubert> initial release
	BUTTONv2-FUN-REQ-095292/A-LIN Message Structure	<jmyslin2 / hzubert> deleted requirements BUTTON-IIR-REQ-107292/C-LIN - ButtonStatusID, BUTTON-IIR-REQ-107293/B-LIN - ButtonStatusValue
	BUTTONv2-SR-REQ-096645/C-LIN BCP message structure	<hzubert> added hint for interfaces having less button slots
	BUTTONv2-SR-REQ-096646/C-LIN BCP message structure usage	<hzubert> added hint for interfaces having less button slots
	BUTTONv2-SR-REQ-096647/C-LIN Multiple stuck buttons in BCP message structure	<hzubert> added hint for interfaces having less button slots
	BUTTON-SR-REQ-293306/A-Button Press reaction time (LIN)	<jmyslin2> created a LIN specific requirement. THERE IS NO CONTENT CHANGE. This requirement used to use CAN version for the LIN section and in the implementation guide changing it on LIN effected it on the CAN implementation guide part so had to create a duplicate requirement
	BUTTON-TMR-REQ-293307/A-T_button_unstuck (LIN)	<jmyslin2> created a LIN specific requirement. THERE IS NO CONTENT CHANGE. This requirement used to use CAN version for the LIN section and in the implementation guide changing it on LIN affected it on the CAN implementation guide part so had to create a duplicate requirement
	BUTTON-SR-REQ-293490/A-Receivers of Held Button Presses (LIN)	<jmyslin2> created a LIN specific requirement. THERE IS NO CONTENT CHANGE. This requirement used to use CAN version for the LIN section and in the implementation guide changing it on LIN affected it on the CAN implementation guide part so had to create a duplicate requirement
	BUTTON-SR-REQ-293493/A-Button Presses with no Held function or Combination with another button press (LIN)	<jmyslin2> created a LIN specific requirement. THERE IS NO CONTENT CHANGE. This requirement used to use CAN version for the LIN section and in the implementation guide changing it on LIN affected it on the CAN implementation guide part so had to create a duplicate requirement
	BUTTON-SR-REQ-293491/A-Buttons with Held Function (LIN)	<jmyslin2> created a LIN specific requirement. THERE IS NO CONTENT CHANGE. This requirement used to use CAN version for the LIN section and in the implementation guide changing it on LIN affected it on the CAN implementation guide part so had to create a duplicate requirement
	BUTTON-SR-REQ-293492/A-Receivers of combination type button presses (LIN)	<jmyslin2> created a LIN specific requirement. THERE IS NO CONTENT CHANGE. This requirement used to use CAN version for the LIN section and in the implementation guide changing it on LIN affected it on the CAN implementation guide part so had to create a duplicate requirement
	BUTTON-SR-REQ-293494/A-Combination Type button press operation (LIN)	<jmyslin2> created a LIN specific requirement. THERE IS NO CONTENT CHANGE. This requirement used to use CAN version for the LIN section and in the implementation guide changing it on LIN affected it on the CAN implementation guide part so had to create a duplicate requirement
	BUTTON-SR-REQ-293495/A-Cancelling RBAP (LIN)	<jmyslin2> added volume as an example. Made the requirement LIN specific since impacts the implementation guide for CAN.
	BUTTON-SR-REQ-293496/A-Cancelling RBAP when change to Standby (LIN)	<jmyslin2> created a LIN specific requirement. THERE IS NO CONTENT CHANGE. This requirement used to use CAN version for the LIN section and in the implementation guide changing it on LIN affected it on the CAN implementation guide part so had to create a duplicate requirement

Table of Contents

1 Architectural Design - LIN Interface

1.1 Button Interface Requirements - LIN

1.1.1 BUTTONv2-IIR-REQ-096644/D-LIN BCP Button Press - Button Interface Requirements

BCP_Button_Press (ICPBtnStateRotary, ICPBtnState…)	Method from the Button Input Client to the receiving modules.	ButtonANameID (ICPBtnID_A) 0x0 - 0xFF see Input Translation Matrix ButtonAActivationState (ICPBtnCoding_A) 0x0 Inactive / Not_Pressed 0x1 Active / Pressed 0x2 ShortEvent / Pressed 0x3 ShortElapsed / Pressed 0x4 LongEvent / Pressed 0x5 Stuck 0xF Idle / Not_Pressed ButtonBNameID (ICPBtnID_B) 0x0 – 0xFF see Input Translation Matrix ButtonBActivationState (ICPBtnCoding_B) 0x0 Inactive / Not_Pressed 0x1 Active / Pressed 0x2 ShortEvent / Pressed 0x3 ShortElapsed / Pressed 0x4 LongEvent / Pressed 0x5 Stuck 0xF Idle / Not_Pressed ButtonCNameID (ICPBtnID_C) 0x0 – 0xFF see Input Translation Matrix ButtonCActivationState (ICPBtnCoding_C) 0x0 Inactive / Not_Pressed 0x1 Active / Pressed 0x2 ShortEvent / Pressed 0x3 ShortElapsed / Pressed 0x4 LongEvent / Pressed 0x5 Stuck 0xF Idle / Not_Pressed ButtonDNameID (ICPBtnID_D) 0x0 – 0xFF see Input Translation Matrix ButtonDActivationState (ICPBtnCoding_D) 0x0 Inactive / Not_Pressed 0x1 Active / Pressed 0x2 ShortEvent / Pressed 0x3 ShortElapsed / Pressed 0x4 LongEvent / Pressed 0x5 Stuck 0xF Idle / Not_Pressed

Hint: maybe new interfaces like e.g. new generation interface, contains less button slots (including related ID and coding slots) e.g. 2. In this case only first number of slots e.g. A and B shall be considered).

1.1.2 BUTTON-IIR-REQ-107291/A-LIN setVolume - Button Interface Requirements

SetVolume (ICPVolumeCmd)	Method for incrementing / decrementing Volume (always used with a rotary knob)	0x0 -30 steps 0x1 -29 steps 0x2 -28 steps … 0x1B -3 steps 0x1C -2 steps 0x1D -1 step (decrement volume) 0x1E Not Pressed / Inactive 0x1F +1 step (increment volume) 0x20 +2 steps 0x21 +3 steps … 0x3A +28 steps 0x3B +29 steps 0x3C +30 steps

1.1.3 BUTTONv2-IIR-REQ-096643/C-LIN setVolume 2 - Button Interface Requirements

SetVolume (ICPVolumeCmd2)	Method for incrementing / decrementing Volume (always used with a rotary knob)	0x0 - 7 steps 0x1 - 6 steps 0x2 - 5 steps 0x3 -4 steps 0x4 -3 steps 0x5 -2 steps 0x6 -1 step (decrement volume) 0x7 Not Pressed / Inactive 0x8 +1 step (increment volume) 0x9 +2 steps 0xA +3 steps 0xB +4 steps 0xC +5 steps 0xD +6 steps 0xE +7 steps

1.1.4 BUTTON-IIR-REQ-095295/E-LIN setRotarySteps - Button Interface Requirements

SetRotarySteps (ICPRotary)	Method for incrementing / decrementing Tune/Rotary Browsing (always used with a rotary knob)	0x0 - 7 steps 0x1 - 6 steps 0x2 - 5 steps 0x3 -4 steps 0x4 -3 steps 0x5 -2 steps 0x6 -1 step 0x7 Not Pressed / Inactive 0x8 +1 step 0x9 +2 steps 0xA +3 steps 0xB +4 steps 0xC +5 steps 0xD +6 steps 0xE +7 steps

1.1.5 BUTTON-IIR-REQ-153850/A-LIN - ApplicationInformation0

ApplicationInformation0 (APINFO0)	Method for error reporting for the volume knob	See LIN Data Link and Physical Layer for further information (Chapter "Ford Standard Error Reporting")

1.1.6 BUTTON-IIR-REQ-153851/A-LIN - ApplicationInformation1

ApplicationInformation1 (APINFO1)	Method for error reporting for the rotary knob	See LIN Data Link and Physical Layer for further information (Chapter "Ford Standard Error Reporting")

1.1.7 BUTTON-IIR-REQ-153852/B-LIN - ApplicationInformation2

ApplicationInformation2 (APINFO2)	Method for error reporting for an under voltage error	See LIN Data Link and Physical Layer for further information (Chapter "Ford Standard Error Reporting")

1.1.8 BUTTON-IIR-REQ-153853/A-LIN - ApplicationInformation3

ApplicationInformation3 (APINFO3)	Method for error reporting for an over voltage error	See LIN Data Link and Physical Layer for further information (Chapter "Ford Standard Error Reporting")

1.1.9 BUTTON-IIR-REQ-153854/A-LIN - ApplicationInformation4

ApplicationInformation4 (APINFO4)	Method for reporting that the slave needs to be configured*	See LIN Data Link and Physical Layer for further information (Chapter "Ford Standard Error Reporting")

*not used for now and is always set to "No Configuration Needed"

1.1.10 BUTTON-IIR-REQ-372239/A-LIN - ApplicationInformation0_NewGeneration

ApplicationInformation0_NewGeneration (APINFO_0_NG)	Method for error reporting for knob faults in general. E.g. covers both, volume knob and rotary, if available	See LIN Data Link and Physical Layer for further information (Chapter "Ford Standard Error Reporting")

1.1.11 BUTTON-IIR-REQ-372241/A-LIN - ApplicationInformation1_NewGeneration

ApplicationInformation1_NewGeneration (APINFO_1_NG)	Method for error reporting for hardwired output fault. E.g. of hardwired buttons	See LIN Data Link and Physical Layer for further information (Chapter "Ford Standard Error Reporting")

1.1.12 BUTTON-IIR-REQ-372242/A-LIN - ApplicationInformation2_NewGeneration

ApplicationInformation2_NewGeneration (APINFO_2_NG)	Method for error reporting for an indicator fault. E.g. LED error of indicators	See LIN Data Link and Physical Layer for further information (Chapter "Ford Standard Error Reporting")

1.1.13 BUTTON-IIR-REQ-372243/A-LIN - ApplicationInformation3_NewGeneration

ApplicationInformation3_NewGeneration (APINFO_3_NG)	Method for error reporting for a circuit fault. E.g. short to ground, short to battery, open circuit	See LIN Data Link and Physical Layer for further information (Chapter "Ford Standard Error Reporting")

1.1.14 BUTTON-IIR-REQ-107294/C-LIN - LINStatus

LINStatus (ICPLINStatus)	Method for additional error reporting	0x0 No detected fault 0x1 Reset (module start after reset) 0x2 Reserved (not used) 0x3 Reserved (not used) 0x4 Data Error 0x5 Checksum Error 0x6 Byte Field Framing Error 0x7 ID Parity Error See LIN Data Link and Physical Layer for further information (Chapter "Ford Standard Error Reporting", respectively SAE J2602 Chapter "Error Field Definition")

1.1.15 BUTTON-IIR-REQ-107295/A-LIN - IlluminationZone

IlluminationZone (DSPIlluZone)	Method for the active zones to illuminate	See LIN Illumination Specifications for further information

1.1.16 BUTTON-IIR-REQ-107296/A-LIN - IlluminationLevel1

IlluminationLevel1 (DSPIlluLvl1)	Method for Value for the 8-bit button backlight PWM generator.	See LIN Illumination Specifications for further information

1.1.17 BUTTON-IIR-REQ-107297/A-LIN - IlluminationLevel2

IlluminationLevel2 (DSPIlluLvl2)	Method for Value for the 8-bit knob backlight PWM generator.	See LIN Illumination specifications for further information

1.1.18 BUTTON-IIR-REQ-107298/A-LIN - PartNumberXxxxx

PartNumberXxxxx (ICPPartNumXxxxx) Ex. ICPPartNumIndex, ICPPartNumData3,…	Method for transferring part number. See LIN Data Link and Physical Layer specification for further information.	See LIN Data Link and Physical Layer specification for further information.

1.1.19 BUTTON-IIR-REQ-107299/B-LIN - ConfigDataXxxx

ConfigDataXxxx (DSPLConfigXxxx) Ex. DSPLConfigIndex, DSPLConfigData2,…	Method for sending configuration data to the LIN slave. NOT USED.	See LIN Data Link and Physical Layer specification for further information.

1.1.20 BUTTON-IIR-REQ-117484/A-LIN - SerialNumberXxxx

SerialNumberXxxx (ICPSrNrDigitXX) Ex. ICPSrNrDigit00, ICPSrNrDigit00, ICPSrNrDigit12,…	Method for transferring serial number of the slave to the master for standard readout. Note: this shall be used equivalent to part number readout.	See LIN Bezel Diagnostics SPSS for further information.

1.1.21 BUTTON-IIR-REQ-366705/A-LIN - ICP_RC

ICP_RC	Method for implementing a rolling counter for having the possibility to detect possible transmission faults of this frame. Note: see "BUTTON-SR-REQ-366885-Rolling Counter behavior" for further information	always 0x0: not supported 0x1 - 0xF

1.1.22 BUTTON-IIR-REQ-366707/A-LIN - DSPL_RC

DSPL_RC	Method for implementing a rolling counter for having the possibility to detect possible transmission faults of this frame. Note: see "BUTTON-SR-REQ-366885-Rolling Counter behavior" for further information	always 0x0: not supported 0x1 - 0xF

1.1.23 BUTTON-IIR-REQ-366710/A-LIN - DSPLIlluIndPTS

DSPLIlluIndPTS	Method for setting status of “push-To-start” indicator. Note: see illumination specification for further information	0x0 Off 0x1 On 0x2 Blinking 0x3 Reserved

1.1.24 BUTTON-IIR-REQ-366711/A-LIN - DSPLIlluIndX

DSPLIlluIndX	Method for setting status of indicator on position X. Note: see illumination specification for further information	0x0 Off 0x1 On 0x2 Blinking 0x3 Reserved

1.1.25 BUTTON-IIR-REQ-366718/A-LIN - DSPLIlluBtnPTS

DSPLIlluBtnPTS	Method for setting illumination status of “push-To-start” button Note: see Illumination specification for further information	0x0 Off 0x1 On

1.1.26 BUTTON-IIR-REQ-366719/A-LIN - DSPLIlluBtnX

DSPLIlluBtnX	Method for setting illumination status of button on position X. Note: see illumination specification for further information	0x0 Off 0x1 On

1.1.27 BUTTON-IIR-REQ-366725/A-LIN - DSPLIlluVolKnob

DSPLIlluVolKnob	Method for setting illumination status of knob inside volume ring. In fact, it is On/Off button. Note: see Illumination specification for further information	0x0 Off 0x1 On

1.1.28 BUTTON-IIR-REQ-366726/A-LIN - DSPLIlluBtnChrome

DSPLIlluBtnChrome	Method for setting illumination status of chrome button(s). Note: see Illumination specification for further information	0x0 Off 0x1 On

1.1.29 BUTTON-IIR-REQ-366729/A-LIN - DSPLIlluPWMIndicatorTargetPTS

DSPLIlluPWMIndicatorTargetPTS	Method for setting brightness target value of "push-to-start" indicator. Note: see illumination specification for further information	10 Bit PWM value (linear or logarithmic) see illumination specification for how to use

1.1.30 BUTTON-IIR-REQ-366730/A-LIN - DSPLIlluPWMBacklightTargetPTS

DSPLIlluPWMBacklightTargetPTS	Method for setting brightness target value of "push-to-start" button backlight. Note: see illumination specification for further information	10 Bit PWM value (linear or logarithmic) see illumination specification for how to use

1.1.31 BUTTON-IIR-REQ-366731/A-LIN - DSPLIlluPWMBacklightTarget

DSPLIlluPWMBacklightTarget	Method for setting brightness target value of button area backlight. Note: see illumination specification for further information	10 Bit PWM value (linear or logarithmic) see illumination specification for how to use

1.1.32 BUTTON-IIR-REQ-366732/A-LIN - DSPLIlluPWMIndicatorTarget

DSPLIlluPWMIndicatorTarget	Method for setting brightness target value of indicator area. Note: see illumination specification for further information	10 Bit PWM value (linear or logarithmic) see illumination specification for how to use

1.1.33 BUTTON-IIR-REQ-366733/A-LIN - DSPLIlluPWMTimerUp

DSPLIlluPWMTimerUp	Method for setting value for timer to reach target brightness value for smooth dimming direction upwards. Note: see illumination specification for further information	4 Bit value see illumination specification for how to use

1.1.34 BUTTON-IIR-REQ-366736/A-LIN - DSPLIlluPWMTimerDown

DSPLIlluPWMTimerDown	Method for setting value for timer to reach target brightness value for smooth dimming direction downwards. Note: see illumination specification for further information	4 Bit value see illumination specification for how to use

1.1.35 BUTTON-IIR-REQ-366738/A-LIN - DSPLIlluDimmingCurveType

DSPLIlluDimmingCurveType	Method for setting how PWM values shall be interpreted). Note: see Illumination specification for further information	0x0 linear 0x1 exponential (e-curve)

1.1.36 BUTTON-IIR-REQ-367005/A-LIN - IlluIndAllocX

IlluIndAllocX (X = 1 to 7)	Method for transferring allocation of available indicators. Means: it contains button ID of related indicator at this position. If none is available at this place, "ICPBtnID_Idle" is used. Hint: That is needed for LIN Master to know which indicators are available on connected LIN slave and on which position to find. This is needed e.g. for setting status of indicators via DSPLIlluIndX. Example: Position 1 is "MAX Defrost" Position 2 is no indicator Position 3 is "Traction Control" Position 4 is no indicator Position 5 is no indicator Position 6 is no indicator Example: this would result in: DSPLIlluInd1 = ICPBtnID_MAXDefr DSPLIlluInd2 = ICPBtnID_Idle DSPLIlluInd3 = ICPBtnID_TracContr	ButtonNameID 0x00 – 0xFF see encoding types of ICPBtnID in LDF

	DSPLIlluInd4 = ICPBtnID_Idle
	DSPLIlluInd5 = ICPBtnID_Idle
	DSPLIlluInd6 = ICPBtnID_Idle	

1.1.37 BUTTON-IIR-REQ-367006/A-LIN - IlluBtnAllocX

IlluBtnAllocX (X = 1 to 8)	Method for transferring allocation of available buttons. Means: it contains button ID of related button at this position. If none is available at this plasce, “ICPBtnID_Idle” is used. Hint: That is needed for LIN Master to know which buttons are available on connected LIN slave and on which position to find. This is needed e.g. for setting status of buttons via DSPLIlluBtnX. Example: Position 1 is “MAX Defrost” Position 2 is no button Position 3 is “Traction Control” Position 4 is no button Position 5 is no button Position 6 is no button Example: this would result in: DSPLIlluBtn1 = ICPBtnID_MAXDefr DSPLIlluBtn2 = ICPBtnID_Idle DSPLIlluBtn3 = ICPBtnID_TracContr DSPLIlluBtn4 = ICPBtnID_Idle DSPLIlluBtn5 = ICPBtnID_Idle DSPLIlluBtn6 = ICPBtnID_Idle	ButtonNameID 0x00 – 0xFF see encoding types of ICPBtnID in LDF

1.1.38 BUTTON-IIR-REQ-367267/A-LIN - ICPIlluSmoothDimmSupp

ICPIlluSmoothDimmSupp	Method for transferring if LIN slave supports smooth dimming strategy. Note: For further information on how smooth dimming works, please refer to illumination specification.	0x0 No 0x1 Yes

1.1.39 BUTTON-IIR-REQ-366723/A-LIN - ICP_CRC8

ICP_CRC8	Method for carrying CRC8 from ICP to master Note: see CRC8 requirements for further information	0x00..0xFF

1.1.40 BUTTON-IIR-REQ-366724/A-LIN - DSPL_CRC8

DSPL_CRC8	Method for carrying CRC8 from master to slave. Note: see CRC8 requirements for further information	0x00..0xFF

1.2 General Requirements

1.2.1 BUTTON-SR-REQ-107308/B-LIN Scheduler turnaround frequency for standard interface

It shall be configurable (e.g. via diagnosis) to send each scheduler a specified number of times (e.g. X, Y, …) and then the next one a specified number of times (e.g. Y). After that it shall begin from start.
X, Y shall be possible to set from 0x0 to 0xF.

Note: This not applies for the configuration scheduler(s).

An example for clarification:
E.g. LIN11: X=0, LIN12: Y=1, LIN13: Z=5
This will result in: LIN11 is not sent, LIN12 will be sent 1 times and then LIN13 will be sent 5 times. After that it begins from start.

LIN11: X=0												
LIN12: Y=1	▨						▨					
LIN13: Z=5		▨	▨	▨	▨	▨		▨	▨	▨	▨	▨

1.2.2 BUTTON-SR-REQ-116454/B-LIN Scheduler turnaround default values for standard interface

LIN11: X=0,
LIN12: Y=1,
LIN13: Z=0

1.2.3 BUTTON-SR-REQ-116455/A-Reset ButtonID after button release

If a button is "released" after "pressed" and an "inactive/Not_Pressed " was sent at least once then the corresponding ICPBtnID/ButtonNameID Byte will be changed to "Idle". If the ButtonID-Code is "Idle" then the appropriate ICPBtnCoding/ButtonActivationState Nibble is a don't care and should always be "Inactive/Not_Pressed".

1.2.4 BUTTON-SR-REQ-107300/A-LimpHome state (Button Transmitter)

If a LIN Server (ex. LIN ICP) reaches the limp-home state it should:
- Activate battery saver mode
- Switch off illumination (if available)

1.2.5 BUTTON-SR-REQ-366885/A-Rolling Counter behavior

The rolling counter shall be incremented by one, in each consecutive corresponding frame. As well, if all values of this frame are the same.
If the rolling counter reaches its maximum value, it shall be reset and begin with default value, again.

Hint: Please keep in mind that 0x0 is never used if CRC is supported! (See "BUTTON-SR-REQ-366886-Rolling Counter default value if CRC is supported")

Hint: Please keep in mind that a value of 0x0 shows that CRC is not supported (BUTTON-SR-REQ-366706/A-Rolling Counter default value if CRC not supported)

Example of a rolling counter supporting CRC:

Startup rolling counter is 1. So, first frame carries 1 in RC. Next frame will have 2, and so on until RC is 15. For next frame the rolling counter is reset to default value. So, it will be 1, again.

Hint: This example assumes "BUTTON-SR-REQ-366886-Rolling Counter default value if CRC is supported" defines default value to 1 and size is four bits.

1.2.6 BUTTON-SR-REQ-366886/A-Rolling Counter default value if CRC is supported

If CRC is supported, default value for rolling counter is 0x1.

1.2.7 BUTTON-SR-REQ-366706/A-Rolling Counter default value if CRC not supported

If CRC is NOT supported, default value for rolling counter is 0x0.

1.2.8 BUTTON-SR-REQ-366887/A-Indicator position allocation

This method allows LIN master to get information about position of each indicator. This is needed since, later on, it only sets status of position on LIN slave.
If master requests "IlluIndAllocX" (X will be e.g. 1 to 6) it shall get related button IDs of each available indicator sorted by position.

The position shall not change.

This can be requested every time master needs this information.

For further information of "IlluIndAllocX" see "BUTTON-IIR-REQ-366711-LIN – DSPLIlluIndX"

1.2.9 BUTTON-SR-REQ-366888/A-Button position allocation

This method allows LIN master to get information about position of each button. This is needed since, later on, it only sets status of position on LIN slave.
If master requests "IlluBtnAllocX" (X will be e.g. 1 to 6) it shall get button IDs of each available button sorted by position.

The position shall not change.

This can be requested every time master needs this information.

For further information on "IlluBtnAllocX" see "BUTTON-IIR-REQ-367006-LIN – IlluBtnAllocX"

1.2.10 BUTTON-SR-REQ-383144/A-Illumination configuration

To dynamically configure illumination allocation (means: which buttons are located at which position for background illumination and which indicators are located at which position) the master shall use "BUTTON-SR-REQ-366888-Button position allocation" and "BUTTON-SR-REQ-366887-Indicator position allocation".

1.2.11 BUTTON-SR-REQ-383145/A-Setup illumination allocation

A master shall read illumination allocation only once, if internal information is not available anymore.

This is usually the case after cold boot (e.g. startup from battery reconnect) or warm boot (e.g. reset) and store this information internally (e.g. in volatile memory like RAM).

Hint: This saves transmission time and means, if LIN slave is changed, master needs to be rebooted, to guarantee that illumination allocation is correct.
However, in any other unintentional case of losing this information, master can request this information at any time.

1.2.12 BUTTON-SR-REQ-366708/A-CRC8 algorithm if CRC is supported

Following look up table is used for 0x83 CRC algorithm:

```
uint8 CalcCRC8(uint8 data[], uint8 len)
```

```
{
uint8 crc = 5; // CRC is non-zero CRC when all data is zero
uint8 tmp;
uint8 i = 0;
// CRC Lookup Table for #0x83 = x^8 +x^2 +x +1 (0x107) <=> (0xe0; 0x1c1)
static uint8 CRC_table_0x83[256] = { // so array is not allocated on stack
0x00, 0x07, 0x0E, 0x09, 0x1C, 0x1B, 0x12, 0x15, 0x38, 0x3F, 0x36, 0x31, 0x24, 0x23, 0x2A, 0x2D,
0x70, 0x77, 0x7E, 0x79, 0x6C, 0x6B, 0x62, 0x65, 0x48, 0x4F, 0x46, 0x41, 0x54, 0x53, 0x5A, 0x5D,
0xE0, 0xE7, 0xEE, 0xE9, 0xFC, 0xFB, 0xF2, 0xF5, 0xD8, 0xDF, 0xD6, 0xD1, 0xC4, 0xC3, 0xCA, 0xCD,
0x90, 0x97, 0x9E, 0x99, 0x8C, 0x8B, 0x82, 0x85, 0xA8, 0xAF, 0xA6, 0xA1, 0xB4, 0xB3, 0xBA, 0xBD,
0xC7, 0xC0, 0xC9, 0xCE, 0xDB, 0xDC, 0xD5, 0xD2, 0xFF, 0xF8, 0xF1, 0xF6, 0xE3, 0xE4, 0xED, 0xEA,
0xB7, 0xB0, 0xB9, 0xBE, 0xAB, 0xAC, 0xA5, 0xA2, 0x8F, 0x88, 0x81, 0x86, 0x93, 0x94, 0x9D, 0x9A,
0x27, 0x20, 0x29, 0x2E, 0x3B, 0x3C, 0x35, 0x32, 0x1F, 0x18, 0x11, 0x16, 0x03, 0x04, 0x0D, 0x0A,
0x57, 0x50, 0x59, 0x5E, 0x4B, 0x4C, 0x45, 0x42, 0x6F, 0x68, 0x61, 0x66, 0x73, 0x74, 0x7D, 0x7A,
0x89, 0x8E, 0x87, 0x80, 0x95, 0x92, 0x9B, 0x9C, 0xB1, 0xB6, 0xBF, 0xB8, 0xAD, 0xAA, 0xA3, 0xA4,
0xF9, 0xFE, 0xF7, 0xF0, 0xE5, 0xE2, 0xEB, 0xEC, 0xC1, 0xC6, 0xCF, 0xC8, 0xDD, 0xDA, 0xD3, 0xD4,
0x69, 0x6E, 0x67, 0x60, 0x75, 0x72, 0x7B, 0x7C, 0x51, 0x56, 0x5F, 0x58, 0x4D, 0x4A, 0x43, 0x44,
0x19, 0x1E, 0x17, 0x10, 0x05, 0x02, 0x0B, 0x0C, 0x21, 0x26, 0x2F, 0x28, 0x3D, 0x3A, 0x33, 0x34,
0x4E, 0x49, 0x40, 0x47, 0x52, 0x55, 0x5C, 0x5B, 0x76, 0x71, 0x78, 0x7F, 0x6A, 0x6D, 0x64, 0x63,
0x3E, 0x39, 0x30, 0x37, 0x22, 0x25, 0x2C, 0x2B, 0x06, 0x01, 0x08, 0x0F, 0x1A, 0x1D, 0x14, 0x13,
0xAE, 0xA9, 0xA0, 0xA7, 0xB2, 0xB5, 0xBC, 0xBB, 0x96, 0x91, 0x98, 0x9F, 0x8A, 0x8D, 0x84, 0x83,
0xDE, 0xD9, 0xD0, 0xD7, 0xC2, 0xC5, 0xCC, 0xCB, 0xE6, 0xE1, 0xE8, 0xEF, 0xFA, 0xFD, 0xF4, 0xF3};


while (i <> len)
{
// XOR datat byte into CRC
tmp = (data[i] ^ crc);
// fetch CRC value from table
crc = CRC_table_0x83[tmp]);
}
return crc;
```

Hint: have a look in “BUTTON-SR-REQ-366886-Rolling Counter default value if CRC is supported”

1.2.13 BUTTON-SR-REQ-366709/A-CRC8 notation

CRC algorithm shall use x^8 +x^2 +x +1 (0x83 in “Koopman” notation) to calculate the 8-bit CRC of the data byte set. It has Hamming Distance of four (HD=4) for 119 data bits.

1.2.14 BUTTON-SR-REQ-366712/A-CRC8 initialization

The CRC shall not return zero when the data set is all zero by initializing the CRC algorithm CRC to five (`uint8 crc = 5;--` as shown in the above algorithm)

1.2.15 BUTTON-SR-REQ-366713/A-CRC8 computation

The CRC shall be computed whenever:

- Any of the data is updated
- OR – whenever the message is transmitted. This option has less CPU load

1.2.16 BUTTON-SR-REQ-366714/A-CRC8 calculation

CRC calculation shall use all bytes before CRC byte (includes rolling counter, as well!)

1.2.17 BUTTON-SR-REQ-366715/A-CRC8 unused bits

Unused bits in the message frame shall be set to 0.

1.2.18 BUTTON-SR-REQ-366716/A-CRC8 unused bytes

Bytes with no signal/data (0x0) shall be used in CRC calculation. I.e. All 7 bytes are used in all CRC calculations.

1.2.19 BUTTON-SR-REQ-366720/A-CRC8 data stream

The data stream fed into the CRC shall be composed of a stream of single byte values.

1.2.20 BUTTON-SR-REQ-366721/A-CRC8 algorithm if CRC is not supported

If CRC is not supported whole CRC8 shall be set to 0x00.

Hint: have a look in "BUTTON-SR-REQ-366706-Rolling Counter default value if CRC not supported"

1.2.21 BUTTON-SR-REQ-366722/A-decision of interface variant

A master can decide on an available response/reaction of "IlluIndAllocX" and "IlluBtnAllocX" if a slave supports new generation interface or standard interface.

Means: a slave supporting only standard interface will give no response/reaction on "IlluIndAllocX" and "IlluBtnAllocX" that are in frames "DSPLIlluIndicationAllocation" and "DSPLIlluButtonAllocation" that are unknown, too.

1.2.22 BUTTON-SR-REQ-372268/A-standard interface

The standard interface covers following:

Schedulers:
LIN11,
LIN12,
LIN13,
LINConfig1,
MRF_schedule,
SRF_schedule

Frames:
ICPBtnState,
DSPLSendSignals,
ICPBtnStateRotary,
DSPLConfigCalibrate,
ICPPartNum,
ICPSerialNum,
MasterReq,
SlaveResp

Signals:
ICP_APINFO_0,
ICP_APINFO_1,
ICP_APINFO_2,
ICP_APINFO_3,
ICP_APINFO_4,
ICPBtnCoding_A,
ICPBtnCoding_B,
ICPBtnCoding_C,
ICPBtnCoding_D,
ICPLINStatus,
ICPRotaryCmd,
ICPVolumeCmd,
ICPVolumeCmd2,
ICPBtnID_A,
ICPBtnID_B,
ICPBtnID_C,
ICPBtnID_D,
ICPPartNumIndex,
ICPPartNumData0,
ICPPartNumData1,

ICPPartNumData2,
ICPPartNumData3,
ICPPartNumData4,
ICPPartNumData5,
ICPSrNrDigit00,
ICPSrNrDigit01,
ICPSrNrDigit02,
ICPSrNrDigit03,
ICPSrNrDigit04,
ICPSrNrDigit05,
ICPSrNrDigit06,
ICPSrNrDigit07,
ICPSrNrDigit08,
ICPSrNrDigit09,
ICPSrNrDigit10,
ICPSrNrDigit11,
ICPSrNrDigit12,
ICPSrNrDigit13,
DSPLIlluZone,
DSPLDimmLvl1,
DSPLDimmLvl2,
DSPLConfigIndex,
DSPLConfigData0,
DSPLConfigData1,
DSPLConfigData2,
DSPLConfigData3,
DSPLConfigData4,
DSPLConfigData5,
DSPLConfigData6,
MasterReqB0,
MasterReqB1,
MasterReqB2,
MasterReqB3,
MasterReqB4,
MasterReqB5,
MasterReqB6,
MasterReqB7,
SlaveRespB0,
SlaveRespB1,
SlaveRespB2,
SlaveRespB3,
SlaveRespB4,
SlaveRespB5,
SlaveRespB6,
SlaveRespB7

1.2.23 <u>BUTTON-SR-REQ-372270/A-new generation interface</u>

The new generation interface covers following:

Schedulers:
LINBtnIndIllu,
LINIlluConfig,
LINConfig1NG,
MRF_schedule,
SRF_schedule

Frames:

ICPBtnStateTwoRC,
DSPLIlluBlocks,
DSPLPWMs,
DSPLIlluIndicationAllocation,
DSPLIlluButtonAllocation,
DSPLConfigCalibrate,
ICPPartNumNG,
ICPSerialNumNG,
MasterReq,
SlaveResp

Signals:
ICP_APINFO_0_NG,
ICP_APINFO_1_NG,
ICP_APINFO_2_NG,
ICP_APINFO_3_NG,
ICP_APINFO_4,
ICPBtnCoding_A,
ICPBtnCoding_B,
ICPLINStatus,
ICPRotaryCmd,
ICPVolumeCmd2,
ICPBtnID_A,
ICPBtnID_B,
ICPIlluSmoothDimmSupp,
Reserved3BitIPC,
ICP_RC,
ICP_CRC8,
DSPLIlluIndPTS,
DSPLIlluInd1,
DSPLIlluInd2,
DSPLIlluInd3,
DSPLIlluInd4,
DSPLIlluInd5,
DSPLIlluInd6,
DSPLIlluInd7,
DSPLIlluBtnPTS,
DSPLIlluBtn1,
DSPLIlluBtn2,
DSPLIlluBtn3,
DSPLIlluBtn4,
DSPLIlluBtn5,
DSPLIlluBtn6,
DSPLIlluBtn7,
DSPLIlluBtn8,
DSPLIlluVolKnob,
DSPLIlluBtnChrome,
Reserved1BitDSPL,
DSPL_RC,
DSPL_CRC8,
DSPLIlluPWMIndicatorTarget,
DSPLIlluPWMBacklightTarget,
DSPLIlluPWMIndicatorTargetPTS,
DSPLIlluPWMBacklightTargetPTS,
DSPLIlluPWMTimerUp,
DSPLIlluPWMTimerDown,
DSPLIlluDimmingCurveType,
Reserved7BitDSPL,
IlluIndAlloc1,

IlluIndAlloc2,
IlluIndAlloc3,
IlluIndAlloc4,
IlluIndAlloc5,
IlluIndAlloc6,
IlluIndAlloc7,
IlluBtnAlloc1,
IlluBtnAlloc2,
IlluBtnAlloc3,
IlluBtnAlloc4,
IlluBtnAlloc5,
IlluBtnAlloc6,
IlluBtnAlloc7,
IlluBtnAlloc8,
ICPPartNumIndex,
ICPPartNumData0,
ICPPartNumData1,
ICPPartNumData2,
ICPPartNumData3,
ICPPartNumData4,
ICPPartNumData5,
ICPSrNrDigit00,
ICPSrNrDigit01,
ICPSrNrDigit02,
ICPSrNrDigit03,
ICPSrNrDigit04,
ICPSrNrDigit05,
ICPSrNrDigit06,
ICPSrNrDigit07,
ICPSrNrDigit08,
ICPSrNrDigit09,
ICPSrNrDigit10,
ICPSrNrDigit11,
ICPSrNrDigit12,
ICPSrNrDigit13,
DSPLConfigIndex,
DSPLConfigData0,
DSPLConfigData1,
DSPLConfigData2,
DSPLConfigData3,
DSPLConfigData4,
DSPLConfigData5,
DSPLConfigData6,
MasterReqB0,
MasterReqB1,
MasterReqB2,
MasterReqB3,
MasterReqB4,
MasterReqB5,
MasterReqB6,
MasterReqB7,
SlaveRespB0,
SlaveRespB1,
SlaveRespB2,
SlaveRespB3,
SlaveRespB4,
SlaveRespB5,
SlaveRespB6,
SlaveRespB7

1.3 BUTTONv2-FUN-REQ-095292/A-LIN Message Structure

1.3.1 LIN - BCP (Button Control Panel) Button Press Message Structure

1.3.1.1 LIN - Infotainment Button Press

1.3.1.1.1 BUTTONv2-SR-REQ-096645/C-LIN BCP message structure

Up to 4 infotainment push button press events can be activated simultaneously by the BCP (**hint**: maybe new interfaces like e.g. new generation interface, contains less button slots (including related ID and coding slots) e.g. 2. In this case only first number of slots e.g. A and B shall be considered). The LIN button press messages are periodic. The basic structure of these messages is shown below.

Note: The LIN ICP will always send the LIN Button Encoding (ex 0x1 Active, 0x2 ShortEvent…). For the receiving module of the LIN ICP button presses whether to use the LIN Button Encoding or the Button Encoding Normalization column depends on how a feature is specified. If the feature SPSS or HMI spec (ex. press and hold timer) uses pressed/not pressed then the Button Encoding Normalization column shall be used. If the SPSS or HMI spec (ex. press and hold timer) uses the LIN Button Encoding (ex 0x1 Active, 0x2 ShortEvent…) then the LIN Button Encoding values shall be used.

Button ID	LIN Button Encoding	Button Encoding Normalization
Button A Activation State	0x0 Inactive	0x0 Not Pressed
	0x1 Active	0x1 Pressed
	0x2 ShortEvent	0x2 Pressed
	0x3 ShortElapsed	0x3 Pressed
	0x4 LongEvent	0x4 Pressed
	0x5 Stuck	0x5 Stuck
Button B Activation State	0x0 Inactive	0x0 Not Pressed
	0x1 Active	0x1 Pressed
	0x2 ShortEvent	0x2 Pressed
	0x3 ShortElapsed	0x3 Pressed
	0x4 LongEvent	0x4 Pressed
	0x5 Stuck	0x5 Stuck
Button C Activation State	0x0 Inactive	0x0 Not Pressed
	0x1 Active	0x1 Pressed
	0x2 ShortEvent	0x2 Pressed
	0x3 ShortElapsed	0x3 Pressed
	0x4 LongEvent	0x4 Pressed
	0x5 Stuck	0x5 Stuck
Button D Activation State	0x0 Inactive	0x0 Not Pressed
	0x1 Active	0x1 Pressed
	0x2 ShortEvent	0x2 Pressed
	0x3 ShortElapsed	0x3 Pressed
	0x4 LongEvent	0x4 Pressed
	0x5 Stuck	0x5 Stuck

Infotainment Button Activation State Coding

BYTE X	BYTE X + 1	BYTE X + 2	BYTE X + 3	BYTE X + 4	BYTE X + 5

ICPBtnID_A (ie Button A Name ID)	ICPBtnID_B (ie Button B Name ID)	ICPBtnID_C (ie Button C Name ID)	ICPBtnID_D (ie Button D Name ID)	ICPBtnCoding_A (ie Button A Activation State)	ICPBtnCoding_B (ie Button B Activation State)	ICPBtnCoding_C (ie Button C Activation State)	ICPBtnCoding_D (ie Button D Activation State)

Note: position of x may vary from scheduler frame to scheduler frame

Infotainment Button Message Structure

1.3.1.1.2 BUTTONv2-SR-REQ-096646/C-LIN BCP message structure usage

When a button is activated it shall encode Button A first and if that position is already being used (ie pressed) shall move to the next Button position in the message. If all 4 buttons (A – D) are being used then any new button inputs shall be ignored until one of the 4 buttons are released.

Once a button press is active for a button ID (A – D) the BCP shall not move that same button press to another button ID location. For example if seek is being pressed and held and is assigned to 'Button C Name ID' it shall not change to 'Button A Name ID' while still being pressed.

The default to set Button Activation States A – D is 'Not Pressed / Inactive' unless there is a button activation event.

Example:
1. Button Module powers up and bus awake
2. Button Module sending: Button A Name ID = Inactive (Idle) AND Button Activation State = Not Pressed
3. User presses button X
4. Button Module sending: Button A Name ID = X AND Button Activation State = Pressed
5. User releases button X
6. Button Module sends: Button A Name ID = X AND Button Activation State = Not Pressed at least once.
7. Button Module sends: Button A Name ID = Inactive (Idle) AND Button Activation State = Not Pressed
8. Button Module continues to send (periodically) Button A Name ID = Inactive (Idle) AND Button Activation State = Not Pressed until the next button press. Any new button press changes Button A Name ID from Inactive (Idle) to the new button value.

Hint: maybe new interfaces like e.g. new generation interface contains less button slots (including related ID and coding slots) e.g. 2. In this case only first number of slots e.g. A and B shall be considered.

1.3.1.1.3 BUTTONv2-SR-REQ-096647/C-LIN Multiple stuck buttons in BCP message structure

If all 4 BCP buttons A – D encoded in the BCP_Button_Press message are determined to be 'stuck' (not including setVolume / setRotarySteps signals – Byte 7 and 8) then one of the 4 button bytes (Button A – D) shall be released so that other buttons can be activated.

Hint: maybe new interfaces like e.g. new generation interface contains less button slots (including related ID and coding slots) e.g. 2. In this case only first number of slots e.g. A and B shall be considered.

1.3.1.2 LIN Signal Functionality (not normalized to Pressed / Not Pressed)

LIN signals are sent out periodically at a pre-defined set of time defined in the LDF. The receiving module may choose to utilize the LIN specific signals such as ShortEvent, ShortElapsed, LongEvent, Stuck to reduce the variability of not knowing when a button was pressed since button press could have occurred +/- (LIN periodic rate) msec from the event.

Also the normalization values of Press/Not_Pressed for encodings can be used (unless noted otherwise) but there could be variability of +/- (LIN periodic rate) msec.

1.3.1.2.1 BUTTON-SR-REQ-096734/B-Active

The Active encoding is set when a LIN button is first pressed.

1.3.1.2.2 BUTTON-SR-REQ-096735/A-ShortElapsed

The ShortElapsed button encoding is set 250 msec after the LIN button press event.

1.3.1.2.3 BUTTON-SR-REQ-096736/C-LongEvent

The LongEvent is set 1.5 seconds after the LIN button press event.

1.3.1.2.4 BUTTON-SR-REQ-096650/B-ShortEvent

The ShortEvent encoding is set if button presses are pressed quicker than ShortElapsed time. This value shall never be overwritten before it is sent out on the bus.

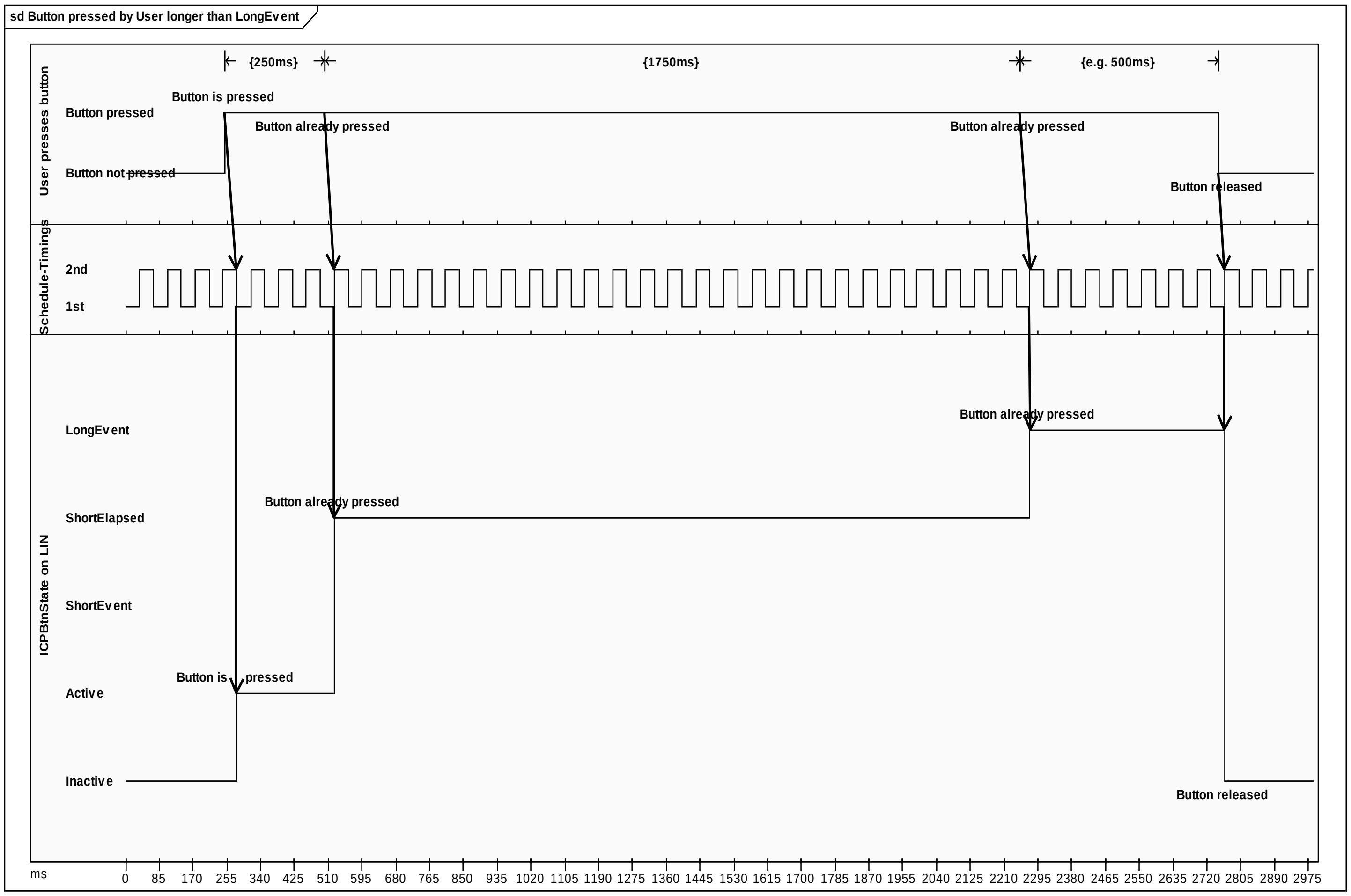
1.3.1.2.5 BUTTON-SR-REQ-096737/B-Stuck

The Stuck encoding is set 120 seconds after a press of a LIN button with no release.

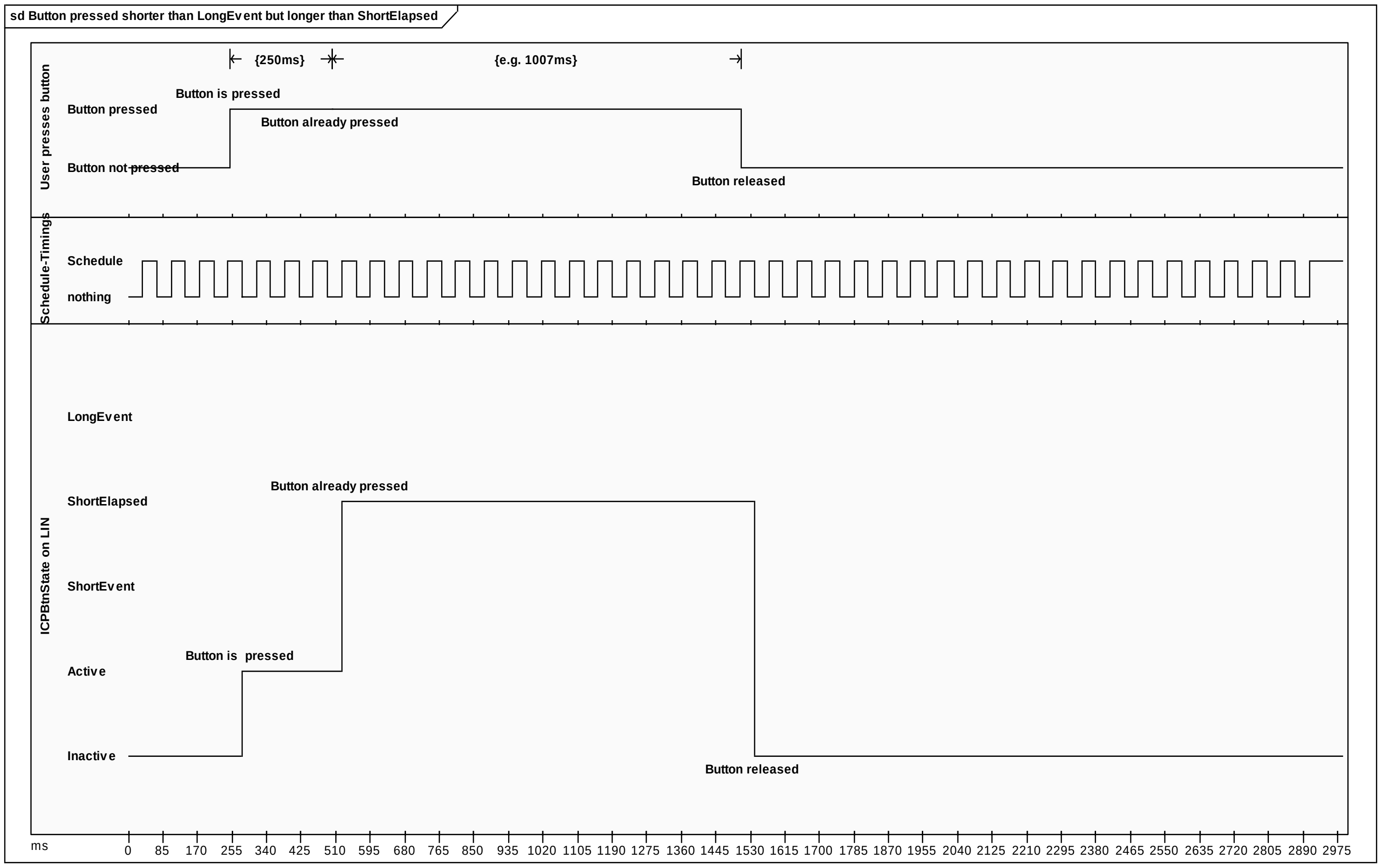
Note: The receiving module of the LIN button press is the module responsible for setting the LIN stuck button DTC as defined in the IDS (infotainment diagnostic specification).

1.3.1.2.6 Button Sequences

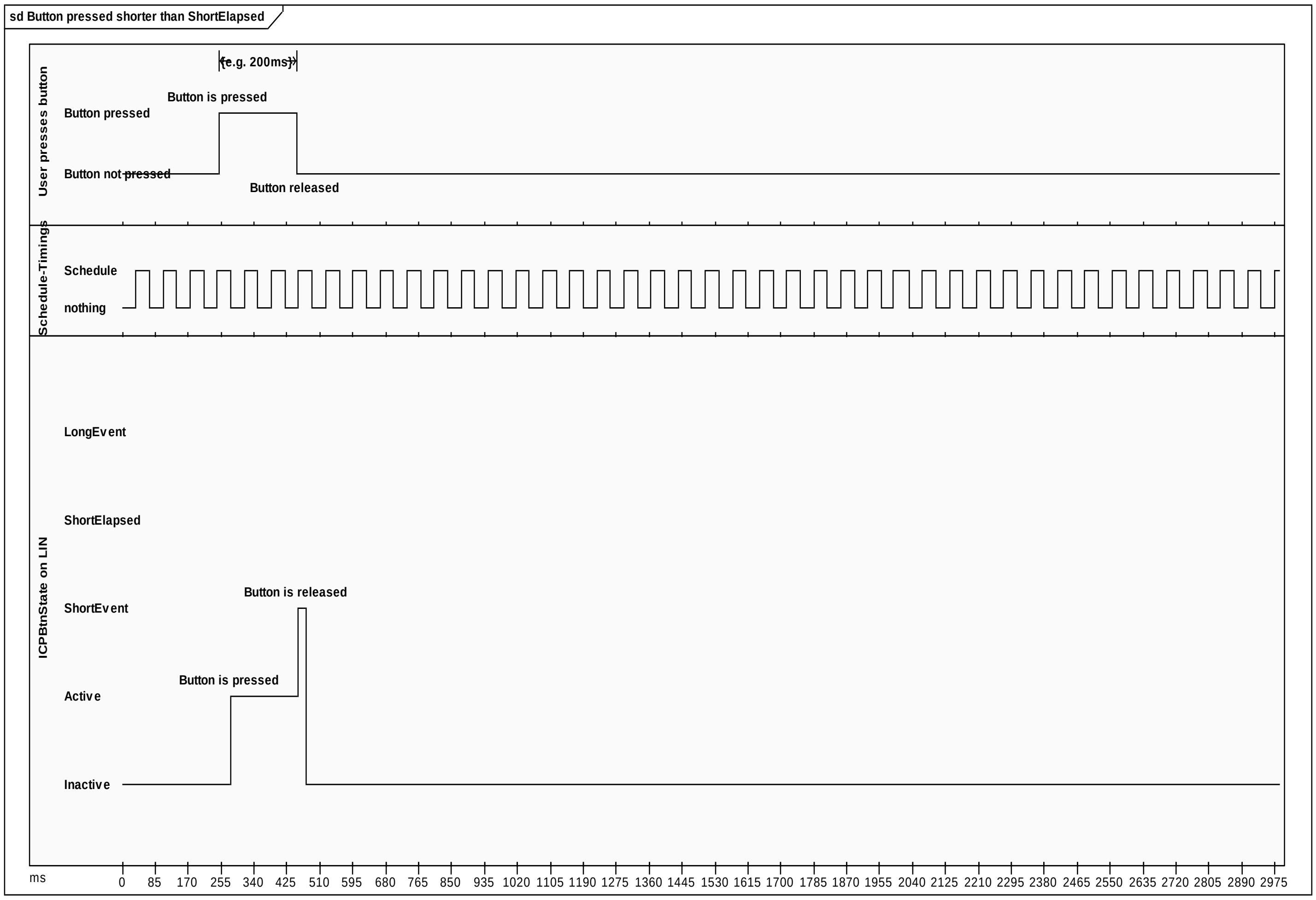
1.3.1.2.6.1 BUTTON-SR-REQ-107301/A-Button pressed by User longer than LongEvent - Button Sequence



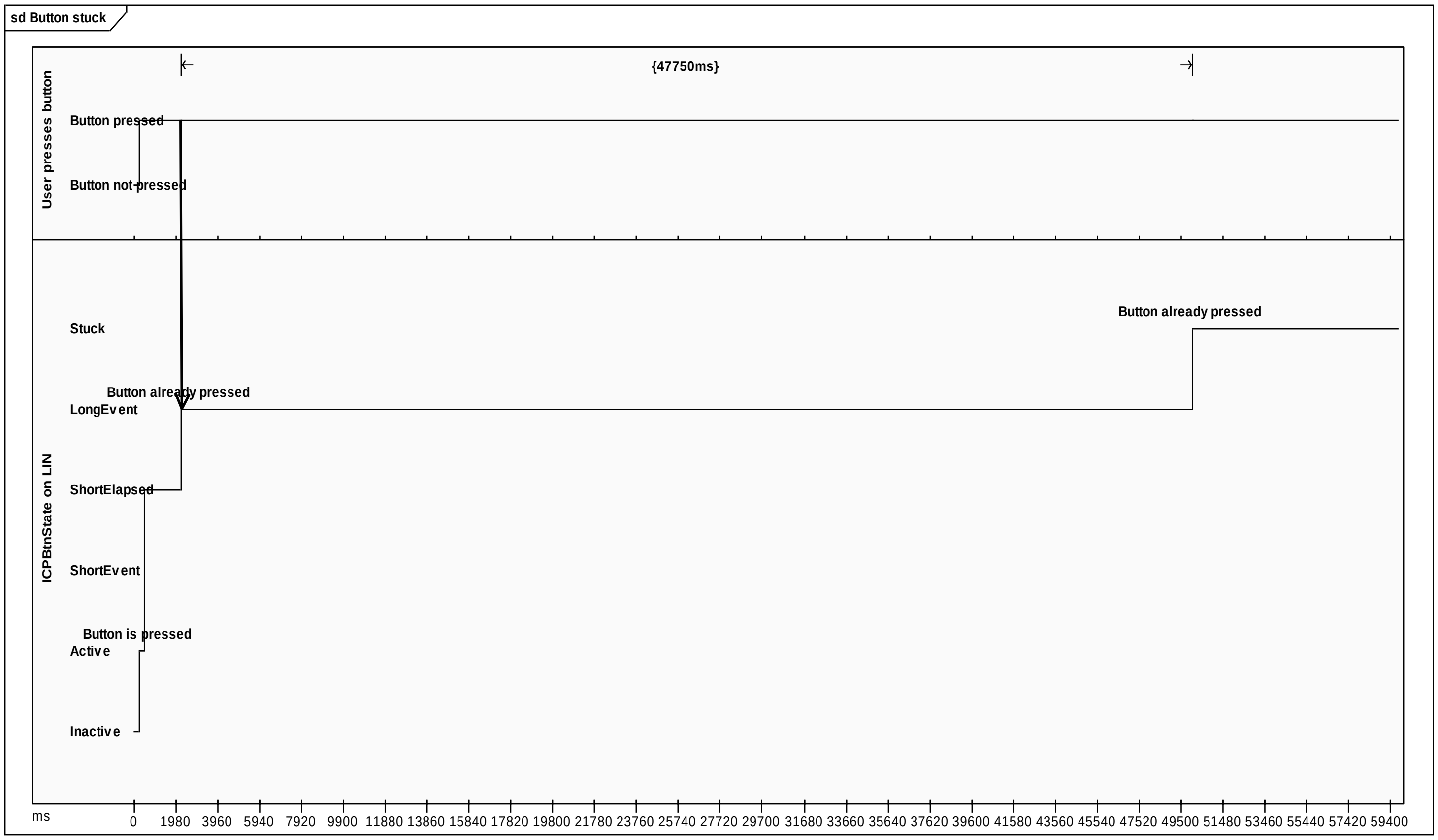
1.3.1.2.6.2 BUTTON-SR-REQ-107302/A-Button Pressed shorter than LongEvent but longer than ShortElapsed - Button Sequence



1.3.1.2.6.3 BUTTON-SR-REQ-107303/A-Button Pressed shorter than ShortElapsed - Button Sequence



1.3.1.2.6.4 BUTTON-SR-REQ-107304/B-Button Stuck - Button Sequence



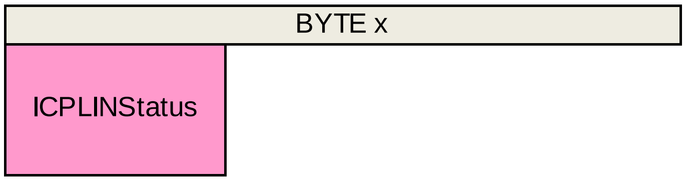
Note: this is an example only and reference the other parts of the SPSS for stuck button timer value to be used if the diagram above does not match what is called out elsewhere in the SPSS. Currently 120 seconds is used for stuck button timer.

1.3.2 ICP Status Transferrring

1.3.2.1 BUTTON-SR-REQ-115765/A-LIN ICP Status Structure

This sub chapter describes how status information is transferred from the ICP to the Master.
ICPLINStatus shows states or errors that are possible to detect but not supported in another way.

An example of the basic structure of this part of message is shown below:
(keep in mind this byte is not entire filled here)



1.3.3 LIN Rotary Structure

1.3.3.1 BUTTON-SR-REQ-116456/A-LIN Rotary Structure

This requirement describes how rotary information is transferred from the ICP to the Master. Only delta counts from frame to frame will be sent.

An example of the basic structure of these parts of messages is shown below:

BYTE x	
ICPVolumeCmd	ICPRotary

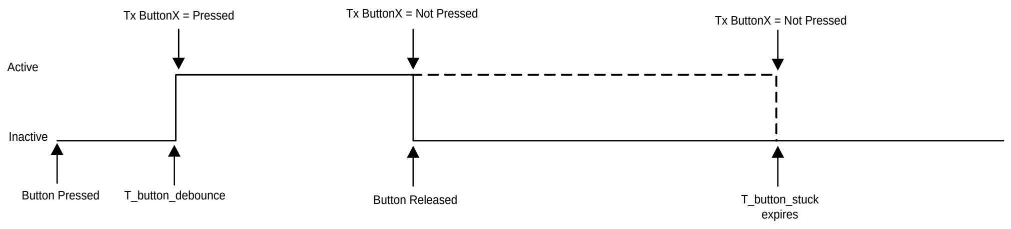
1.4 BUTTONv2-CLD-REQ-095293/A-Button Input Client (Button Transmitter) - LIN

The following sections define the Button Activation Strategy from the Transmitters perspective.

1.4.1 Push Button Activation - Transmitter Timing Requirements

1.4.1.1 BUTTONv2-SR-REQ-107305/A-Transmitter Button Activation Process timing (LIN)

The TBAP timing figure below will always remain true for any button activation event using the Pressed/Not Pressed normalization (ie not using LIN button sequences). The exception to this rule is for "Rotary Knobs" which is covered in the next section.



Transmitter Button Activation Process (TBAP) Timing

1.4.1.2 BUTTONv2-TMR-REQ-096739/B-T_reaction_time (LIN)

Name	Description	Units	Range	Resolution	Default
T_reaction_time (LIN)	The maximum transmitter reaction time from when a push button switch is closed until the push button message is put on the LIN bus.	msec	0-1000	10	70

1.4.2 Push / Touch Button Activation - Transmitter Functional Requirements

1.4.2.1 BUTTONv2-SR-REQ-107306/A-Button Pressed / Not Pressed transimission (LIN)

Once a button is pressed and debounced on the transmitter the button message will be sent to the receiver with the button signal coding set to "Pressed" (ie LongEvent, shortEvent…).

Once the button is released the transmitter will set the button coding as "Not Pressed / Inactive".

1.4.2.2 BUTTON-SR-REQ-293306/A-Button Press reaction time (LIN)

The transmitter (ex.EFP, ICP, SWC, SDM) reaction time from when a push button switch is closed until the push button message is put on the bus shall not exceed T_reaction_time.

Note: this does not apply to the touch sense buttons. Reference applicable specifications for touch sense debounce requirements.

1.4.2.3 BUTTONv2-SR-REQ-103693/A-Transmitter Stuck Button (LIN)

When the transmitter determines a button to be stuck:

-- The transmitter will keep the button encoding status as ‘Stuck' as long as the button is stuck. Upon a new ignition cycle the button encoding shall remain as 'Stuck' until the button is determined to be operational.
-- Once a button becomes unstuck after previously being stuck then after remaining unstuck for T_button_unstuck the button shall become operational again.

1.4.2.4 BUTTON-TMR-REQ-293307/A-T_button_unstuck (LIN)

Name	Description	Units	Range	Resolution	Default
T_button_unstuck (LIN)	Once a button is determined to be stuck then T_button_unstuck is the time the button has to be unstuck before the button can be operational again. Note: always use the default value	sec	0-100	1	10

1.4.3 setRotarySteps signal (LIN)

1.4.3.1 BUTTON-SR-REQ-095296/D-Tx SetRotarySteps (LIN)

The setRotarySteps signal could possibly be used to increment / decrement for the tune or fast browse function for example. Reference the applicable SPSS section for details. Each setRotarySteps step encoding shall be treated as a press event in the remainder of this document.

The Button Input Client (Button Transmitter) shall increment the setRotarySteps signal by 1 for every detected rotary knob detent in the clockwise direction.

- For example if the Button Transmitter detects 3 detents in the clockwise direction could send +3 (fast turn) or three +1 button press messages (slower turn) as long as no information is lost. Note for LIN do not need to see the Not_Press to act on one of the volume steps.

The Button Input Client (Button Transmitter) shall decrement the setRotarySteps signal by 1 for every detected rotary knob detent in the counter-clockwise direction.

- For example if the Button Transmitter detects 3 detents in the counter clockwise direction could send -3 (fast turn) or three -1 button press messages (slower turn) as long as no information is lost. Note for LIN do not need to see the Not_Press to act on one of the volume steps.

The Button Input Client (Button Transmitter) shall send out the delta counts accumulated since the last setRotarySteps signal sent out on the bus.

If delta counts are 0 then “Not Pressed / Inactive” shall be sent out.

1.4.4 setVolume signal (LIN)

1.4.4.1 BUTTON-SR-REQ-096743/B-Tx SetVolume (LIN)

The setVolume signal can be used to increment / decrement the Volume for a volume rotary knob for example. Reference the applicable SPSS section for details. Each setVolume step encoding shall be treated as a press event in the remainder of this document.

The Button Input Client (Button Transmitter) shall increment the setVolume signal by 1 for every detected rotary knob detent in the clockwise direction.

- For example if the Button Transmitter detects 3 detents in the clockwise direction could send +3 (fast turn) or three +1 button press messages (slower turn) as long as no information is lost. Note for LIN do not need to see the Not_Press to act on one of the volume steps.

The Button Input Client (Button Transmitter) shall decrement the setVolume signal by 1 for every detected rotary knob detent in the counter-clockwise direction.

- For example if the Button Transmitter detects 3 detents in the counter clockwise direction could send -3 (fast turn) or three -1 button press messages (slower turn) as long as no information is lost. Note for LIN do not need to see the Not_Press to act on one of the volume steps.

The Button Input Client (Button Transmitter) shall send out the delta counts accumulated since the last setVolume signal sent out on the bus.

If delta counts are 0 then “Not Pressed / Inactive” shall be sent out.

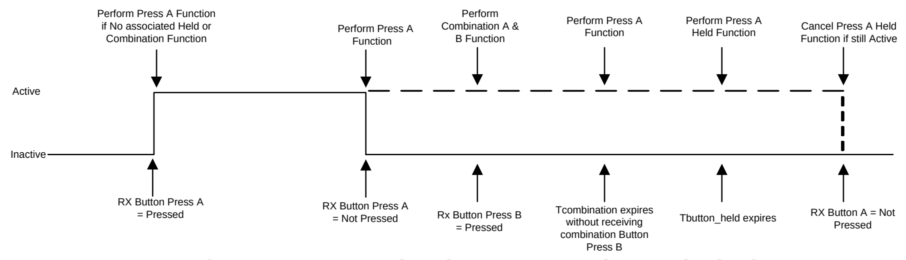
1.5 BUTTONv2-CLD-REQ-095294/A-Button Input Server (Button Receiver) - LIN

The following sections define the Button Activation Strategy from the Receivers perspective.

1.5.1 Button Activation - Receiver Timing Requirements (LIN)

1.5.1.1 BUTTONv2-SR-REQ-096745/B-Receiver Button Activation Process (RBAP) Timing (LIN)

All component receivers of button press information shall implement the Receiver Button Activation Process (RBAP) timing defined in the figure below.


Perform Press A Function if No associated Held or Combination Function
Perform Press A Function
Perform Combination A & B Function
Perform Press A Function
Perform Press A Held Function
Cancel Press A Held Function if still Active
Active
Inactive
RX Button Press A = Pressed
RX Button Press A = Not Pressed
Rx Button Press B = Pressed
Tcombination expires without receiving combination Button Press B
Tbutton_held expires
RX Button A = Not Pressed

Receiver Button Activation Process (RBAP) Timing

Exception: If a particular button supports press and hold function there may be times where a functional requirement may require the function to first be performed on the press (not wait for not pressed) and then take additional action when Tbutton_held expires and the buttons are determined to be held. This should only be performed if explicitly called out in a functional requirement otherwise the function shall be performed only on a Not Press or when Tbutton_held expires as shown in the RBAP above.

Exception 2: Rotary Knobs signals do not need to wait for the not pressed (ex setVolume, setRotarySteps).

1.5.2 Button Activation - Receiver Functional Requirements (LIN)

1.5.2.1 BUTTONv2-SR-REQ-096746/A-Receivers of Button Presses follow RBAP (LIN)

All receivers of button press information shall activate the RBAP upon receipt of the button press message.

1.5.2.2 BUTTONv2-SR-REQ-096747/B-Button Receiver Sampling Rate (LIN)

The sampling rate used by the receiver to read incoming button information messages shall be fast enough to read the multiple incoming messages.

1.5.2.3 BUTTON-SR-REQ-293490/A-Receivers of Held Button Presses (LIN)

The receiver shall determine whether the specific button has an associated held function.

1.5.2.4 BUTTON-SR-REQ-293493/A-Button Presses with no Held function or Combination with another button press (LIN)

If the button does not have an associated held or combination type function, then the receiver shall perform the function associated with a button press immediately upon receipt of the button press message and the RBAP process shall be exited.

1.5.2.5 BUTTON-SR-REQ-293491/A-Buttons with Held Function (LIN)

If the button does have an associated held function, the receiver shall start a hold timer (Tbutton_held) and wait for button 'not pressed' information.

-- If a button 'not pressed' message is not received prior to the expiration of Tbutton_held, the receiver shall perform the associated held function for that button press.

-- If a button 'not pressed' message is received prior to the expiration of Tbutton_held, then the receiver shall perform the associated press function.

1.5.2.6 BUTTON-SR-REQ-293492/A-Receivers of combination type button presses (LIN)

The receiver shall determine whether the button press has an associated combination type button press.

1.5.2.7 BUTTON-SR-REQ-293494/A-Combination Type button press operation (LIN)

When the receiver detects a button press (A) that has an associated combination the receiver must wait Tcombination before executing the button press function associated with the single button press (A).

-- If a second button (B) is received prior to Tcombination expiring, the receiving module must now determine if this (B) is part of a valid combination with (A).
- If the combination is valid, then the resulting combination (A + B) can be performed.
- If the combination is invalid, then no combination function is performed and the button presses (A) & (B) can be processed independently if allowed by the receiving module.

-- If Tcombination expires, then button press (A) is now valid and the (A) function can be performed.

1.5.2.8 BUTTON-SR-REQ-293495/A-Cancelling RBAP (LIN)

The receiver shall cancel the RBAP (Receiver Button Activation Process) upon:

- Receipt of the button 'Not Pressed' message.
- If the receiver does not receive a button 'Not Pressed' message within T_RBAP_Timeout
  - Unless noted otherwise all buttons shall timeout at some point in case of a stuck button
  - T_RBAP_Timeout may vary depending on how the button is used for a particular feature. The T_RBAP_Timeout would be part of error handling for a particular feature unless a T_RBAP_Timeout value is explicitly noted in a feature SPSS.
    - For example: if a feature does not have a press and hold or combination button press associated with it then that button might time out quickly. Other features like Seek/Volume have press and hold features associated with them and would not time out for a longer period of time. Follow up with the Ford HMI or Ford feature team on what make sense for timeout values.

Reference:
- For cancelling Volume Button RBAP reference requirements "VOL-TMR_REQ-292290-T_Vol_RBAP_Timeout" and "VOL-REQ-292289-Volume Press and Hold Timeout".

1.5.2.9 BUTTON-SR-REQ-293496/A-Cancelling RBAP when change to Standby (LIN)

For Infotainment receivers any event that shall cause a transition from Functional mode to Standby mode shall cancel a RBAP unless noted otherwise. The receiver shall cancel the operation, if active, and perform the required actions to enter Standby State.

1.5.2.10 BUTTONv2-SR-REQ-096749/A-Receivers of SetVolume Button presses (LIN)

The Button Input Server (Volume Setting Server) shall increment/decrement the volume based on the delta count of the volume steps received since the last SetVolume signal update.

1.5.2.11 BUTTON-SR-REQ-107307/A-Receivers of SetRotarySteps Button Presses (LIN)

The Button Input Server (Rotary Setting Server) shall increment/decrement the steps based on the delta count of the steps received since the last SetRotarySteps signal update.

2 Appendix: Reference Documents

Reference #	Document Title
1	Input Translation Matrix
2	LIN DVM specifications
3	LIN Data LINK and Physical Layer specifications
4	LIN Physical Layer Approved Components
5	SAE J2602-3
6	SAE J2602-2
7	SAE J2602-1
8	LIN Database File (LDF)
9	LIN Illumination Specification
10	LIN Netcom SOW
11	
12	
13	