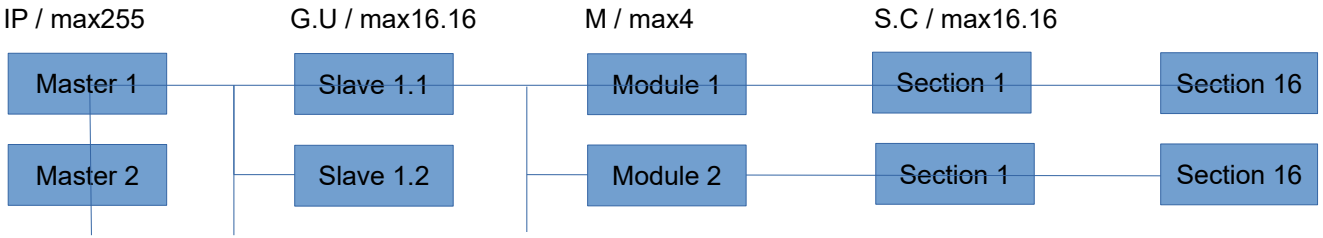
PROTOKÓŁ TRANSMISJI DANYCH PBMS2019 Ver. 2.0

Protokół transmisji danych w systemie Pyrobox nie został z góry zaprojektowany lecz powstawał stopniowo (był poszerzany i modyfikowany z zachowaniem kompatybilności w dół) wraz z rozwojem systemu więc niektóre jego założenia i elementy są konsekwencją takiego właśnie procesu.

Sposób adresowania w systemie obwodów zapalczych (cue) także był modyfikowany i obecnie adresujemy je przez wskazanie kolejno:

G / group	zakres 1..16	grupa urządzeń slave wykonujących określone (podobne) zadania
U / unit	zakres 1..16	identyfikator urządzenia slave w grupie
M / module	zakres 1..4	numer modułu wykonawczego kontrolowanego przez urządzenie slave
S / section	zakres 1..16	numer sekcji w module wykonawczym
C / channel	zakres 1..16	numer kanału w sekcji

Każdy program/kontroler master może więc nadzorować maksymalnie 16 grup urządzeń slave.
W każdej grupie może być maksymalnie 16 identyfikowanych urządzeń slave lub dowolna ich ilość jeśli zrezygnujemy z ich identyfikowania (co oznacza brak możliwości indywidualnego odwołania się do danego urządzenia np. zdalnego testowania obwodów zapalczych).
Każde urządzenie slave może nadzorować (obecnie) maksymalnie 4 moduły wykonawcze.
Moduł wykonawczy może mieć maksymalnie 16 sekcji, a każda sekcja ma 16 kanałów.



Urządzeniem slave może być:
- komputer z odpowiednim oprogramowaniem
- dedykowany konwerter np. konwerter/hab PB12CH4L1 obsługujący do 4 modułów
- moduł wykonawczy slave np. 32-kanałowy moduł SM032

Transmisja danych w sieci komunikacyjnej Pyrobox odbywa się w standardzie RS485.
Parametry transmisji: baudrate = 9600, bytesize = 8, parity = no, stopbits = 1, softflow = false, hardflow = false.
W transmisji mogą być wykorzystywane standardowe elementy sieci RS485 np. repeater, hub, radiomodem (transmisja przezroczysta).

Dane przesyłane są w postaci ciągu znaków ASCII.
Ciąg znaków zawiera kolejno:
- znak „{” oznaczający początek pakietu danych (start)
- 4 znaki sumy kontrolnej CRC16 CCITT X-MODEM, oznaczane dalej jako <crc>
- 2 znaki literowe oznaczające komendę/rozkaz, dalej <cmd>
- 2 znaki z zakresu 00..FF wskazujące część adresu obwodu zapalczego Group.Unit , dalej <adr>
- 2 znaki z zakresu 00..03 wskazujące kolejną część adresu Module, dalej <mod>
- 2 znaki z zakresu 00..FF wskazujące kolejną część adresu Section.Channel, dalej <cue>
- n znaków danych sformatowanych w sposób przedstawiony w tabelach poniżej
- znak „}” oznaczający koniec pakietu (stop)

W przypadku niektórych komend część z wymienionych elementów ciągu nie jest wykorzystywana.

Komendy podzielone zostały na 3 grupy:
- grupa 1 - rozkazy kierowane jednocześnie do wszystkich urządzeń
- grupa 2 - rozkazy kierowane do jednego lub wszystkich urządzeń w grupie
- grupa 3 - rozkazy kierowane tylko do jednego urządzenia w grupie

GROUP 1 / All Units, part 1

COMMANDS:		1	2	3	4	5	6	7	8	9	10	11	chars	string	answer
MN	Arm_On_All	{	<crc>	MN	}								8	{D29AMN)	
MN	Arm_On_Unit_PBMS2019	{	<crc>	MN	<adr>	}							10		
MF	Arm_Off_All	{	<crc>	MF	}								8	(5392MF}	
MF	Arm_Off_Unit_PBMS2019	{	<crc>	MF	<adr>	}							10		
MR	Reset_All	{	<crc>	MR	}								8	{0127MR}	
MR	Reset_Unit_PBMS2019	{	<crc>	MR	<adr>	}							10		
XP	HeartBeat	{	<crc>	XP	}								8	{DDE3XP}	
XM	Start (sfx: time)	{	<crc>	XM	}								8	{1E7FXM}	
XT	Message (sfx: txt=str)	{	<crc>	XT	<txt>	}							max29		
FA	All_Slave_Units (sfx: all=xxx)	{	<crc>	FA	<all>	}							11		
MX	**SAM_Start_PBMS2019v2.0**	**{**	**<crc>**	**MX**	**<hms>**	**}**							**14**		
MY	**SAM_Stop_PBMS2019v2.0**	**{**	**<crc>**	**MY**	**}**								**8**	**{B04CMY}**	
MZ	**SAM_HeartBeat_PBMS2019v2.0**	**{**	**<crc>**	**MZ**	**}**								**8**	**{802FMZ}**	
WA	**Warning_Ready_PBMS2019v2.0**	**{**	**<crc>**	**WA**	**}**								**8**	**{CFCDWA}**	
WB	**Warning_Armed_PBMS2019v2.0**	**{**	**<crc>**	**WB**	**}**								**8**	**{FFAEWB}**	
WC	**Warning_Disarmed_PBMS2019v2.0**	**{**	**<crc>**	**WC**	**}**								**8**	**{EF8FWC}**	
WD	**Warning_Safe_PBMS2019v2.0**	**{**	**<crc>**	**WD**	**}**								**8**	**{9F68WD}**	

MN	uzbroić system / uzbroić urządzenie <adr>			
MF	rozbroić system / rozbroić urządzenie <adr>			
MR	zresetować system / zresetować urządzenie <adr>			
XP	sygnał heartbeat			
XM	sygnał start	→ funkcja: **time**		
XT	wiadomość tekstowa	→ funkcja: **txt=str**	→ <txt>max21 / x..z	← x..z = max 21 chars
FA	wykonać zadanie xxx ze skryptu **ASU**	→ funkcja: **all=xxx**	→ <all>3 / xxx	← xxx = 001...999 number
MX	sygnał start w trybie SAM		→ <hms>6 / hhmmss	← hh=00..23 „hour” / mm=00..59 „minute” / ss=00..59 „second”
MY	awaryjny sygnał stop w trybie SAM		→ Death Man Switch	
MZ	sygnał heartbeat w trybie SAM		→ max every 4.5sec	
WA	pokazać że urządzenie jest w gotowości / manual		→ yellow signal ligth	
WB	pokazać że urządzenie jest uzbrojone / auto 1sec after MN		→ red signal light	
WC	pokazać że urządzenie jest rozbrojone / auto 1sec after MF		→ green signal light	
WD	pokazać że urządzenie jest bezpieczne / manual		→ blue signal light	

GROUP 1 / All Units, part 2 PBMS2019

COMMANDS:		1	2	3	4	5	6	7	8	9	10	11	chars	string	answer
	SERVICE MODE ONLY:														
PS	Write_IDNO	{	<crc>	PS	<ser>	}							17		yes
PS	Write_IDNO_check	{	<crc>	PS	<ser>	}							17		
	NORMAL MODE:														
PX	Read_IDNO	{	<crc>	PX	}								8	{D542PX}	yes
PX	Read_IDNO_answer	{	<crc>	PX	<ser>	}							17		
PY	Read_USER	{	<crc>	PY	<ser>	}							17		yes
PY	Read_USER_answer	{	<crc>	PY	<ser>	<usr>	}						26		
PU	Write_USER	{	<crc>	PU	<ser>	<usr>	}						26		yes
PU	Write_USER_check	{	<crc>	PU	<ser>	<usr>	}						26		
PZ	Read_ADDR	{	<crc>	PZ	<ser>	}							17		yes
PZ	Read_ADDR_answer	{	<crc>	PZ	<ser>	<add>	}						23		
PA	Write_ADDR	{	<crc>	PA	<ser>	<add>	}						23		yes
PA	Write_ADDR_check	{	<crc>	PA	<ser>	<add>	}						23		
PD	Read_DATA	{	<crc>	PD	<ser>	}							17		yes
PD	Read_DATA_answer	{	<crc>	PD	<inf>	<pwr>	<ser>	<usr>	<add>	<arm>	<ant>	}	max46		

PS	zapisać numer seryjny urządzenia	→ <ser>9 / xxxxyynnn	← xxxx=ABcd „type” / yy=19..99 „year” / nnn=001..999 „ser#”
PX	przesłać numer seryjny urządzenia		
PY	przesłać identyfikator urządzenia	→ <usr>9 / xxxyyynnn	← xxx=ABC „user” / yyy=ABC „tool” / nnn=001..999 „idn#”
PU	zapisać identyfikator urządzenia		
PZ	przesłać adres konfiguracyjny urządzenia	→ <add>6 / xgumsc	← x=2..5 / g=0..F / u=0..F / m=0..3 / s=0..F / c=0,4,8,C
PA	zapisać adres konfiguracyjny urządzenia		
PD	przesłać pełną informację nt. Urządzenia	→ <inf>4 / xxxx	← x=0,1,2,3,7,B,F number of sections 0,2,3,4,8,12,16
		→ <pwr>6 / xxxyyy	← xxx=000..360 pwr#1 inside / yyy=000..500 pwr#2 outside
		→ <arm>1 / x = 0..3	← 0=safe / hard_arm=0/1 / soft_arm=0/2 / 3=armed
		→ <ant>3 / xxx = 000..100	← % RF signal power

GROUP 2 / All Or One Unit In Group

COMMANDS:		1	2	3	4	5	6	7	8	9	10	11	chars	string	answer
FC	Fire_Cue_G	{	<crc>	FC	<adr>	<mod>	<cue>	}					14		
FO	Fire_Cue_U (sfx: one=nn)	{	<crc>	FO	<adr>	<mod>	<cue>	<one>	}				15		
FP	Fire_Pulse_G (sfx: imp=xxx)	{	<crc>	FP	<adr>	<mod>	<cue>	<pls>	}				17		
FP	Fire_Pulse_U (sfx: imp=xxx/nn)	{	<crc>	FP	<adr>	<mod>	<cue>	<pls>	<one>	}			18		
FS	Regular_Sequencer_G (sfx: seq=aa/bb)	{	<crc>	FS	<adr>	<mod>	<cue>	<ssn>	<sst>	}			18		
FS	Regular_Sequencer_U (sfx: seq=aa/bb/nn)	{	<crc>	FS	<adr>	<mod>	<cue>	<ssn>	<sst>	<one>	}		19		
FH	Speed_Up_Sequencer_G (sfx: shi=aa/bb/cc)	{	<crc>	FH	<adr>	<mod>	<cue>	<ssn>	<sst>	<ssd>	}		20		
FH	Speed_Up_Sequencer_U (sfx: shi=aa/bb/cc/nn)	{	<crc>	FH	<adr>	<mod>	<cue>	<ssn>	<sst>	<ssd>	<one>	}	21		
FL	Speed_Down_Sequencer_G (sfx: slo=aa/bb/cc)	{	<crc>	FL	<adr>	<mod>	<cue>	<ssn>	<sst>	<ssd>	}		20		
FL	Speed_Down_Sequencer_U (sfx: slo=aa/bb/cc/nn)	{	<crc>	FL	<adr>	<mod>	<cue>	<ssn>	<sst>	<ssd>	<one>	}	21		
FT	Programming_Sequencer_G (sfx: sdf=xxx)	{	<crc>	FT	<adr>	<sdf>	}						14		
FT	Programming_Sequencer_U (sfx: sdf=xxx/nn)	{	<crc>	FT	<adr>	<sdf>	<one>	}					14		
FN	Zero_Delay_Firing_G (sfx: nxt=x, x=2..5)	{	<crc>	FN	<adr>	<mod>	<cue>	..	<adr5>	<mod5>	<cue5>	}	max38		
MS	Set_Select_Cues_G (sfx: set=xxxxxxxx)	{	<crc>	MS	<adr>	<mod>	<set>	}					20		
MS	Set_Select_Cues_U (sfx: set=xxxxxxxx/nn)	{	<crc>	MS	<adr>	<mod>	<set>	<one>	}				21		

FC	odpalić cue w grupie		
FO	odpalić cue w urządzeniu <one>	→ funkcja: **one=nn**	→ <one>2 / nn = 01..16 unit#
FP	odpalić cue impulsem xxx	→ funkcja: **imp=xxx/nn**	→ <pls>3 / xxx = 100..900 msec ++ nn = 01..16 „one”
FS	uruchomić sekwencję	→ funkcja: **seq=aa/bb/nn**	→ <ssn>2 / aa = 02..99 ++ <sst>2 / bb = 02..99 ++ nn = 01..16 „one”
FH	uruchomić sekwencję step-up	→ funkcja: **shi=aa/bb/cc/nn**	→ <ssn>2 / aa = 02..99 ++ <sst>2 / bb = 02..99 ++ <ssd>2 / cc = 01..49 ++ nn = 01..16
FL	uruchomić sekwencję step-down	→ funkcja: **slo=aa/bb/cc/nn**	→ <ssn>2 / aa = 02..99 ++ <sst>2 / bb = 02..99 ++ <ssd>2 / cc = 01..49 ++ nn = 01..16
FT	uruchomić sekwencję ze skryptu **SDF**	→ funkcja: **sdf=xxx/nn**	→ <sdf>3 / xxx = 001..999 ++ nn = 01..16 „one”
FN	odpalić x cue w grupie	→ funkcja: **nxt=x**	→ x = 2..5 repeats <adr><mod><cue>
MS	włączyć y cue w urządzeniu	→ funkcja: **set=xxxxxxxx/nn**	→ <set>8 / xxxxxxxx = 00000000..FFFFFFFF ++ nn = 01..16 „one”

GROUP 3 / One Unit In Group, part 1

COMMANDS:		1	2	3	4	5	6	7	8	9	10	11	chars	string	answer
XY	Master_Slave_Link_Test	{	<crc>	XY	<adr>	}							10		yes
XY	MSLT_answer	{	<crc>	XY	<adr>	<inf>	<pwr>	}					14/20		
XY	MSLT_answer_PBMS2019	{	<crc>	YX	<adr>	<ser>	}						19		
YX	Slave_Master_Link_Test	{	<crc>	YX	<adr>	}							10		yes
YX	SMLT_answer	{	<crc>	YX	<adr>	}							10		
YX	Slave_Master_Link_Test_PBMS2019	{	<crc>	YX	<adr>	<ser>	}						19		**yes**
YX	SMLT_answer_PBMS2019	{	<crc>	YX	<adr>	<ser>	}						19		
XZ	Master_Reserve_Link_Test	{	<crc>	XZ	}								8	{7CA9XZ}	yes
XZ	MRLT_answer	{	<crc>	XZ	}								8	{7CA9XZ}	
XZ	MRLT_answer_PBMS2019	{	<crc>	XZ	<ser>	}							17		
ZX	Reserve_Master_Link_Test	{	<crc>	ZX	}								8	{3A89ZX}	yes
ZX	RMLT_answer	{	<crc>	ZX	}								8	{3A89ZX}	
ZX	RMLT_answer_PBMS2019	{	<crc>	ZX	<ser>	}							17		
ZZ	Master_Reserve_Switch	{	<crc>	ZZ	}								8	(1ACBZZ}	

XY sprawdzić połączenie master → slave <adr>
YX sprawdzić połączenie slave <adr> → master
XZ sprawdzić połączenie master → master_reserve
ZX sprawdzić połączenie master_reserve → master
ZZ przekazać kontrolę master → reserve master

GROUP 3 / One Unit In Group, part 2 PBMS2019v2.0

COMMANDS:		1	2	3	4	5	6	7	8	9	10	11	chars	string	answer
XK	**Master_Keypad_Link_Test**	**{**	**<crc>**	**XK**	**}**								**8**	**{7EB9XK}**	**yes**
XK	**MKLT_answer**	**{**	**<crc>**	**XK**	**}**								**8**	**{7EB9XK}**	
XK	**MKLT_answer_PBMS2019v2.0**	**{**	**<crc>**	**XK**	**<ser>**	**}**							**17**		
KX	**Keypad_Master_Link_Test**	**{**	**<crc>**	**KX**	**}**								**8**	**{0ACBKX}**	**yes**
KX	**KMLT_answer**	**{**	**<crc>**	**KX**	**}**								**8**	**{0ACBKX}**	
KX	**KMLT_answer_PBM2019v2.0**	**{**	**<crc>**	**KX**	**<ser>**	**}**							**17**		
XV	**Master_Starter_Link_Test**	**{**	**<crc>**	**XV**	**}**								**8**	**{DB25XV}**	**yes**
XV	**MSLT_answer**	**{**	**<crc>**	**XV**	**}**								**8**	**{DB25XV}**	
XV	**MSLT_answer_PBMS2019v2.0**	**{**	**<crc>**	**XV**	**<ser>**	**}**							**17**		
VX	**Starter_Master_Link_Test**	**{**	**<crc>**	**VX**	**}**								**8**	**{7FE4VX}**	**yes**
VX	**SMLT_answer**	**{**	**<crc>**	**VX**	**}**								**8**	**{7FE4VX}**	
VX	**SMLT_answer_PBMS2019v2.0**	**{**	**<crc>**	**VX**	**<ser>**	**}**							**17**		

XK sprawdzić połączenie master → keypad
KX sprawdzić połączenie keypad → master
XV sprawdzić połączenie master → starter
VX sprawdzić połączenie starter → master

GROUP 3 / One Unit In Group, part 3

COMMANDS:		1	2	3	4	5	6	7	8	9	10	11	chars	string	answer
XA	Master_Player_Link_Test	{	<crc>	XA	}								8	{DFF3XA}	yes
XA	MPLT_answer	{	<crc>	XA	}								8	{DFF3XA}	
XA	MPLT_answer_PBMS2019	{	<crc>	XA	<ser>	}							17		
AX	Player_Master_Link_Test	{	<crc>	AX	}								8	{E500AX}	yes
AX	PMLT_answer	{	<crc>	AX	}								8	{E500AX}	
AX	Player_Master_Link_Test_PBMS2019	{	<crc>	AX	<ser>	}							17		yes
AX	PMLT_answer_PBMS2019	{	<crc>	AX	<ser>	}							17		
XB	Master_Player_Reserve_Link_Test	{	<crc>	XB	}								8	{EF90XB}	yes
XB	MPRLT_answer	{	<crc>	XB	}								8	{EF90XB}	
XB	MPRLT_answer_PBMS2019	{	<crc>	XB	<ser>	}							17		
BX	Player_Reserve_Master_Link_Test	{	<crc>	BX	}								8	{B053BX}	yes
BX	PRMLT_answer	{	<crc>	BX	}								8	{B053BX}	
BX	Player_Reserve_Master_Link_Test_PBMS2019	{	<crc>	BX	<ser>	}							17		yes
BX	PRMLT_answer_PBMS2019	{	<crc>	BX	<ser>	}							17		

XA sprawdzić połączenie master → player
AX sprawdzić połączenie player → master
XB sprawdzić połączenie master → reserve player
BX sprawdzić połączenie reserve player → master

GROUP 3 / One Unit In Group, part 4

COMMANDS:		1	2	3	4	5	6	7	8	9	10	11	chars	string	answer
TC	Test_Cue	{	<crc>	TC	<adr>	<mod>	<cue>	}					14		
TS	Test_Section	{	<crc>	TS	<adr>	<mod>	<cue>	}					14		
TM	Test_Module	{	<crc>	TM	<adr>	<mod>	<cue>	}					14		
SC	Send_Test_Cue	{	<crc>	SC	<adr>	}							10		yes
SC	STC_answer	{	<crc>	SC	<adr>	<0/1>	}						11		
SS	Send_Test_Section	{	<crc>	SS	<adr>	}							10		yes
SS	STS_answer	{	<crc>	SS	<adr>	<tds>	}						14		
SM	Send_Test_Module	{	<crc>	SM	<adr>	}							10		yes
SM	STM_answer	{	<crc>	SM	<adr>	<tdp>	<tdm>	}					1..4x28		

TC	wykonać test cue	
TS	wykonać test całej sekcji	
TM	wykonać test całego modułu	
SC	przesłać wyniki testu cue	→ <0/1>1 = 0/1
SS	przesłać wyniki testu sekcji	→ <tds>4 = 0000..FFFF
SM	przesłać wyniki testu modułu	→ <tdp>2 = 00..11 packet# / <tdm>16 = 0000000000000000..FFFFFFFFFFFFFFFF