SRS

XMPS-2000

May-23 Task SRS_v1.1

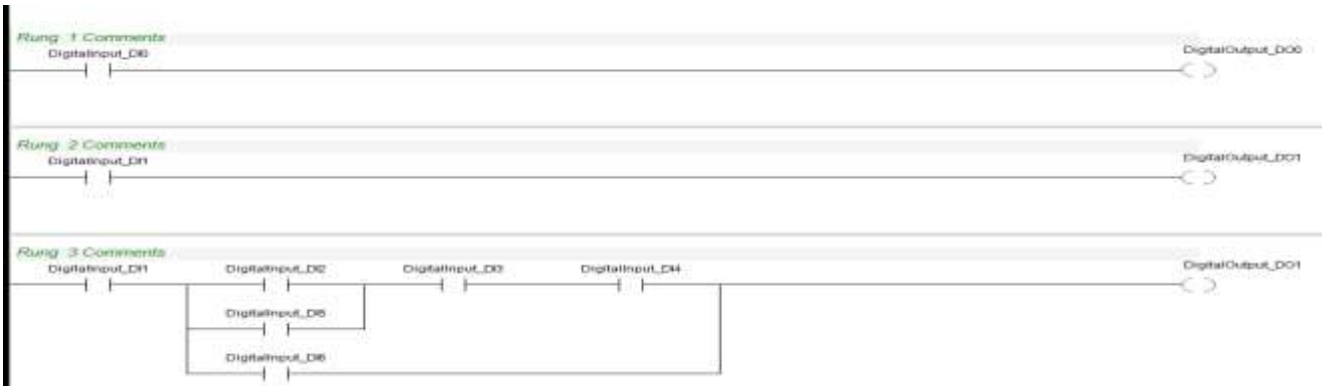
Index

Sr. No	Description	Page no.
1	Objectives	2
2	MCode	2
3	CCode	17
4	Retentive Timer	33
5	Generate Disabling variable	34
6	Cuser tool tip feature	

Objectives

In this Software requirements specification (SRS) we are implementing MCode and CCode to remove the firmware side Interpreter process, conversion of App.csv to MCode and Config.csv to CCode. We will directly send MCode and CCode to the XM14DT/XM17ADT via Modbus TCP or TFTP.

This document describes the MCode and CCode Interpreter implementation requirements. It will explain the code format with exist XMPS2000 project.

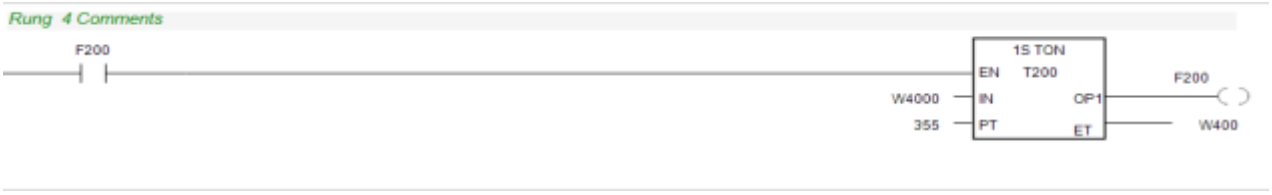


1. MCODE

- Convert .csv to MCODE:

  Eg.—

  Main Program



- Ex-1 : Example for creating MCode:
- In this example we follow the given steps to convert csv file to MCode format (step 1,2,3 are only for reference).
- Refer step 4,6,7,8 steps for MCode details.

1. Current XMPS200 generate the following App.csv file

Line No	T/C Name	Output Type	Data Type	Enable	Output1	Output2	Op Code	Input1	Input2	Input 3	Input 4	Comments	Window Name
1	-	0	0	-	Q0:000.00	-	90	I1:000.00	-	-	-		B01
2	-	0	0	-	Q0:000.01	-	90	I1:000.01	-	-	-		
3	-	0	0	-	D10:0000	-	10	I1:000.02	I1:000.05	-	-		
4	-	0	0	-	D10:0001	-	0	D10:0000	I1:000.03	I1:000.04	-		
5	-	0	0	-	D10:0002	-	10	D10:0001	I1:000.06	-	-		
6	-	0	0	-	Q0:000.01	-	0	I1:000.01	D10:0002	-	-		
7	T200	2	6	F2:000	F2:000	W4:000	01D6	F2:001	355	-	-		

For this equivalent csv (current logic in PLC) we convert char form data,

As shown in below buff (*max 192 char in 1 rung)

(This is for reference only Flairminds can make own logic also)

-1st rung

lcCVS_Logic_Buff - char [192]				
[0...99]	char [100]	0x2002f6a0		
	lcCVS_Logic_Buff[0]	char	49 '1'	Rung no.
	lcCVS_Logic_Buff[1]	char	44 ','	
	lcCVS_Logic_Buff[2]	char	45 '-'	t/c name
	lcCVS_Logic_Buff[3]	char	44 ','	
	lcCVS_Logic_Buff[4]	char	48 '0'	Output type
	lcCVS_Logic_Buff[5]	char	48 '0'	
	lcCVS_Logic_Buff[6]	char	44 ','	
	lcCVS_Logic_Buff[7]	char	48 '0'	Enable
	lcCVS_Logic_Buff[8]	char	48 '0'	
	lcCVS_Logic_Buff[9]	char	48 '0'	
	lcCVS_Logic_Buff[10]	char	48 '0'	
	lcCVS_Logic_Buff[11]	char	44 ','	
	lcCVS_Logic_Buff[12]	char	45 '-'	operandtype1
	lcCVS_Logic_Buff[13]	char	44 ','	
	lcCVS_Logic_Buff[14]	char	81 'Q'	OUTPUT 1
	lcCVS_Logic_Buff[15]	char	48 '0'	
	lcCVS_Logic_Buff[16]	char	58 ':'	
	lcCVS_Logic_Buff[17]	char	48 '0'	

	lcCVS_Logic_Buff[18]	char	48 '0'	
	lcCVS_Logic_Buff[19]	char	48 '0'	
	lcCVS_Logic_Buff[20]	char	46 '.'	
	lcCVS_Logic_Buff[21]	char	48 '0'	
	lcCVS_Logic_Buff[22]	char	48 '0'	
	lcCVS_Logic_Buff[23]	char	44 ','	
	lcCVS_Logic_Buff[24]	char	45 '-'	OUTPUT 2
	lcCVS_Logic_Buff[25]	char	44 ','	
	lcCVS_Logic_Buff[26]	char	48 '0'	
	lcCVS_Logic_Buff[27]	char	48 '0'	
	lcCVS_Logic_Buff[28]	char	57 '9'	OPCODE
	lcCVS_Logic_Buff[29]	char	48 '0'	
	lcCVS_Logic_Buff[30]	char	44 ','	
	lcCVS_Logic_Buff[31]	char	73 'I'	
	lcCVS_Logic_Buff[32]	char	49 '1'	
	lcCVS_Logic_Buff[33]	char	58 ':'	
	lcCVS_Logic_Buff[34]	char	48 '0'	
	lcCVS_Logic_Buff[35]	char	48 '0'	INPUT -1
	lcCVS_Logic_Buff[36]	char	48 '0'	
	lcCVS_Logic_Buff[37]	char	46 '.'	
	lcCVS_Logic_Buff[38]	char	48 '0'	
	lcCVS_Logic_Buff[39]	char	48 '0'	
	lcCVS_Logic_Buff[40]	char	44 ','	
	lcCVS_Logic_Buff[41]	char	45 '-'	INPUT -2
	lcCVS_Logic_Buff[42]	char	44 ','	
	lcCVS_Logic_Buff[43]	char	45 '-'	INPUT -3
	lcCVS_Logic_Buff[44]	char	44 ','	
	lcCVS_Logic_Buff[45]	char	45 '-'	INPUT -4
	lcCVS_Logic_Buff[46]	char	44 ','	
	lcCVS_Logic_Buff[47]	char	44 ','	
	lcCVS_Logic_Buff[48]	char	66 'B'	
	lcCVS_Logic_Buff[49]	char	48 '0'	
	lcCVS_Logic_Buff[50]	char	49 '1'	
	lcCVS_Logic_Buff[51]	char	13 '\r'	
	lcCVS_Logic_Buff[52]	char	10 '\n'	

2. This equivalent code will convert into Messung “MCode” format using firmware interpreter logic.
   (Need MCODE data in below format)

3. Messung MCode format for 1 Rung is:

1	Rung No.	4	byte
2	Data type	4	byte
3	Enable Type	4	byte
4	Enable	4	byte
5	OPCODE	4	byte
6	Type of operand 1	4	byte
7	OP1	4	byte
8	Type of operand 2	4	byte
9	OP2	4	byte
10	Type of operand 3	4	byte
11	OP3	4	byte
12	Type of operand 4	4	byte
13	OP4	4	byte
14	No. of Operand	4	byte
15	T_C Name	4	byte
16	Output 1	4	byte
17	Output 2	4	byte

4. Ex1 equivalent C code conversion example for MCode buffer creation:

Rung 1---
I1:000.00 MOV Q0:000.00
Rung 2---
I1:000.01 MOV Q0:000.01
Rung 3---
D10:000= I1:000.02 OR I1:000.05
Rung 4---
D10:001= D10:000 AND I1:000.03 AND I1:000.04
Rung 5---
D10:002= D10:001 OR I1:000.06
Rung 6---
Q0:000.01= I1:000.01 AND D10:002

5. For MCode with example App.csv, Please refer the MCODE.xlsx for more details

6. MCode for above Ex1-(decimal values for address)

*Address may be in decimal or hex

	SOF	$	
	Total no. of rungs		

	Description	Actual MCode Buffer values	Comments
1	Rung Number	1	
2	Data type	0	
3	Enable type	0	
4	Enable	0	
5	OPCODE	9	MOV operation
6	Type_operand_1	0	
7	OP1	574111744	
8	Type_operand_2	2	
9	OP2	0	
10	Type_operand_3	2	
11	OP3	0	
12	Type_operand_4	2	
13	OP4	0	
14	No of Operand_	1	
15	T_C Name	0	
16	Output1,	574095360	
17	Output2	0	
18	Rung Number	2	
19	Data type	0	
20	Enable type	0	
21	Enable	0	
22	OPCODE	9	MOV operation
23	Type_operand_1	0	
24	OP1	574111748	
25	Type_operand_2	2	
26	OP2	0	
27	Type_operand_3	2	
28	OP3	0	
29	Type_operand_4	2	
30	OP4	0	
31	No of Operand_	1	
32	T_C Name	0	

33	Output1,	574095364	
34	Output2	0	
35	Rung Number	3	
36	Data type	0	
37	Enable type	0	
38	Enable	0	
39	OPCODE	1	OR
40	Type_operand_1	0	
41	OP1	574111752	
42	Type_operand_2	0	
43	OP2	574111764	
44	Type_operand_3	2	
45	OP3	0	
46	Type_operand_4	2	
47	OP4	0	
48	No of Operand_	2	
49	T _C Name	0	
50	Output1,	574244864	
51	Output2	0	
52	Rung Number	4	
53	Data type	0	
54	Enable type	0	
55	Enable	0	
56	OPCODE	0	AND
57	Type_operand_1	0	
58	OP1	574244864	
59	Type_operand_2	0	
60	OP2	574111756	
61	Type_operand_3	0	
62	OP3	574111760	
63	Type_operand_4	2	
64	OP4	0	
65	No of Operand_	3	
66	T_C Name	0	
67	Output1,	574244868	
68	Output2	0	
69	Rung Number	5	
70	Data type	0	
71	Enable type	0	
72	Enable	0	
73	OPCODE	1	OR
74	Type_operand_1	0	
75	OP1	574244868	

76	Type_operand_2	0	
77	OP2	574111768	
78	Type_operand_3	2	
79	OP3	0	
80	Type_operand_4	2	
81	OP4	0	
82	No of Operand_	2	
83	T_C Name	0	
84	Output1,	574244872	
85	Output2	0	
86	Rung Number	6	
87	Data type	0	
88	Enable type	0	
89	Enable	0	
90	OPCODE	0	AND
91	Type_operand_1	0	
92	OP1	574111748	
93	Type_operand_2	0	
94	OP2	574244872	
95	Type_operand_3	2	
96	OP3	0	
97	Type_operand_4	2	
98	OP4	0	
99	No of Operand_	2	
100	T_C Name	0	
101	Output1,	574095364	
102	Output2	0	
103	Rung Number	7	
104	Data type	6	
105	Enable type	1	enable check
106	Enable	574128128	
107	OPCODE	27	0.01s Ton
108	Type_operand_1	0	
109	OP1	574128132	
110	Type_operand_2	3	
111	OP2	250	Numeric i/p
112	Type_operand_3	2	
113	OP3	0	
114	Type_operand_4	2	
115	OP4	0	
116	No of Operand_	2	
117	T_C Name	0	
118	Output1,	574128128	

119	Output2	536987168	

7. Before first rung and after last rung there will be some addition like below.

Before first Rung-

Description			
SOF	$	4	byte
No. of Rungs	MAX 1000	4	byte

After Last Rung-

Description			
EOF	#	4	byte

5. MCode details.

5.1 SOF- Start of Frame ---

We have defined "$" as a start of MCode buffer.
Size - 4 byte

5.2 No. of Rungs---

No. of rungs will be calculate after compilation of code of total no. of MCode rungs.

Size-4 bytes

5.3 Rung no.---

Rung no. ---Rung no as per sequential flow of program
Size- 4 bytes

5.4 Data Type---

As per Opcode – (defined earlier same as per XMPS-100)
Size- 4 bytes

OPCODE	Data Type
0	BOOL
1	BYTE
2	WORD
3	D WORD
4	INT
5	REAL
6	TON
7	TOFF
8	CTU
9	CTD
A	TP

5.5 Enable type---

This is applicable when FB with enable is used by user.
Size- 4 byte

Enable value	Enable Type
0	Unchecked
1	checked
2	Negation checked

5.6 Enable—

Here actual MCU address will come…. (I1, Q0, D3 and F2 etc.)

For value 1- Physical I/P, O/P bit address or memory bit address (F2:000 AND D3:000) .

For value 2- Physical I/P, O/P bit address memory bit address (F2:000 AND D3:000).

During compilation/ generate code command each address will convert into MCU address. (Check section xxx for more details)

Size- 4 bytes

5.7 Opcode---

This is the Opcode of Rung operation as per below table.
Size- 4 bytes

OPCODE	OPCODE operation
0	AND
1	OR
2	XOR
3	NOT
4	ADD
5	SUB
6	MUL
7	DIV
8	MOD
9	MOV
A	SHL
B	SHR
C	ROR
D	ROL
10	LIMIT
11	GT

12	GE
13	LT
14	LE
15	EQ
16	NE
17	Rising Edge
18	Falling Edge
19	CTU
1A	CTD
1B	0.01S TON
1C	0.1TON
1D	1s TON
1E	0.01S TOFF
1F	0.1TOFF
20	1s TOFF
21	0.01s TP
22	0.1s TP
23	1s TP
24	RS
25	SR
26	Scalling1

5.8 Type of operand 1---

Normal, Negation or Numeric—for logical instructions
Normal, Numeric—for other instructions

Operand value	Type operand
0	Normal mode
1	Negation
2	not used
3	numeric

Size- 4 byte

5.9 OP1---

Operand 1

Here actual MCU address will come….(after conversion of I1,Q0,F2,W4,D3,P5 etc) or numeric value (as per selected data type).

During compilation/ generate code command each address will convert into MCU address. (Check section xxx for more details)

Size- 4 bytes

5.10 Type of operand 2---

Normal, Negation or Numeric—for logical instructions
Normal, Numeric—for other instructions

Operand value	Type operand
0	Normal mode
1	Negation
2	not used
3	numeric

Size- 4 byte

5.11OP2---

Operand 2

Here actual MCU address will come….(after conversion of I1,Q0,F2,W4,D3,P5 etc) or numeric value (as per selected data type).

During compilation/ generate code command each address will convert into MCU address. (Check section xxx for more details)

Size- 4 bytes

5.12 Type of operand 3---

Normal, Negation or Numeric—for logical instructions
Normal, Numeric—for other instructions

Operand value	Type operand
0	Normal mode
1	Negation
2	not used
3	numeric

Size-4 byte

5.13 OP3---

Operand 3

Here actual MCU address will come….(after conversion of I1,Q0,F2,W4,D3,P5 etc) or numeric value (as per selected data type).

During compilation/ generate code command each address will convert into MCU address. (Check section xxx for more details)

Size- 4 bytes

5.14 Type of operand 4---

Normal, Negation or Numeric—for logical instructions
Normal, Numeric—for other instructions

Operand value	Type operand
0	Normal mode
1	Negation
2	not used
3	numeric

Size- 4 byte

5.15 OP4---

Operand 4

Here actual MCU address will come….(after conversion of I1,Q0,F2,W4,D3,P5 etc) or numeric value (as per selected data type).

During compilation/ generate code command each address will convert into MCU address. (Check section xxx for more details)

Size- 4 bytes

5.16 No. of Operand---

Size 4-bytes

We have given max 4 address for any instruction.

User can use greater than 1 address for any instruction.

So for particular rung how many operands are used that count should be here.

5.17 T_C Name---

Size 4-bytes

T/C value	T/C Type
0	Not used/default
0-49	0.01s Ton
50-199	0.1s Ton
200-255	1s Ton
0-49	0.01s Toff
50-199	0.1s Toff
200-255	1s Toff
0-49	0.01s TP
50-199	0.1s TP
200-255	1s TP

Timer Counter No.

We have limit the Max 255 all types of Timers & Max 255 all types of counters.

So according to this it will increase. And do not repeat.

5.18 Output1---

Here actual MCU address will come….(after conversion of I1,Q0,F2 ,P5 etc.)

During compilation/ generate code command each address will convert into MCU address. (Check section xxx for more details)

Size- 4 bytes

5.19 Output2---

Here actual MCU address will come….(after conversion of I1,Q0,F2,T6,C7,W4 etc.)

During compilation/ generate code command each address will convert into MCU address. (Check section xxx for more details)

This is right now applicable for Time 7 Counter instructions only.

Size- 4 bytes

5.20 EOF---

End of frame—"#"

After generation of MCode for all backend rungs at last this character should add.

Size-4 byte

CPU address to MCU address conversion idea & base address:

For address conversion logic please refer below table for Base Addresses.

Block No	*Type*	*Logical Address Range*	*Total nos PLC address*	*MCU Address starting Address*	*Next address ---*
0	Output address	Q0:000 to Q0:255	256	0X2001 C000	Plus 2
		Q0:000.00…Q0:000.15 to Q0:255.00…Q0:255.15	4096	0x2238 0000	Plus 4
1	Input address	I1:000 to I1:255	256	0X2001 C200	Plus 2
		I1:000.00…I1:000.15 to I1:255.00…I1:255.15	4096	0X2238 4000	Plus 4
2	Flags (Memory bits)	F2.000 to F2.255	256	0x2238 8000	Plus 4
3	Status	S3:000 to S3:255	256	0x2001 C420	Plus 2

4	Integer Word	W4:000 to W4:255	256	0x2001 C620	Plus 2
5	Floating Point	P5:000 to P5:255	256	0x2001 C820	Plus 4
6	Timers	T6:000 to T6:255	256	0x2001 CC20	Plus 2
7	Counters	C7:000 to C7:255	256	0x2001 CE20	Plus 2
8	Reserved for future	X8.000 to X8.255	256	0x2001 D020	Plus 2
9	Reserved for future	Y9.000 to Y9.255	256	0x2001 D040	Plus 2
10	Auto memory flag	D10:000 to D10:2048	2048	0x223A8400	Plus 4

2. CCode (PLC configuration settings)

We need byte by byte data of all PLC configuration settings.

We will download the same using TFTP or TCP same as per the Application CCode data.

Configuration interpreter will interpret the PLC configuration settings in below format:

Data Sequence	Text	Decimals
SOF	#	
COM Settings	!A	8513
Ethernet Settings	!B	8514
nEthernet settings	!C	8515
PLC model	!D	8516
Remote IO Configuration	!E	8517
PLC on board IO Configuration	!F	8518
Modbus Type	!G	8519
Modbus RTU	!H	8520
Modbus TCP Client	!I	8521
Modbus TCP Server	!J	8522
Retentive	!K	8523
Initial	!L	8524
Expansion	!M	8525
Universal I/O	!N	8526
CRC	!O	8527
EOF	&	

CRC- need to find crc calculation

Expansion UIUO

We need to update the UIUO mode value start from 1(we are using 0 as no value)

Update		
Value	UI Mode	UO Mode
1	Digital	Digital
2	0-10V	0-10V
3	0-5V	0-20mA
4	0-20mA	4-20mA
5	4-20mA	
6	Resistance	
7	PT100	
8	PT1000	
9	10K NTC	
10	20K NTC	

Example: For CCode with example config.csv, Please refer the CCODE.xlsx for more details

This is only example with config.csv files.

We need all data in decimal format only.

We need only Data column. (Comments are given for understanding of data.)

CCode Frame Format-Configuration.csv file

Byte no.	Data	Comment	Data Size
0	#	SOF	1 Byte
1	!A (8513 Decimal)	Com settings	2 Byte
2	0 to 4	Baud rate	1 Byte
3	0 to 3	parity	1 Byte
4	0-65535	Communication timeout	2 Byte
5			
6	0-9	No. of retries	1 Byte
7	6-8	Data length	1 Byte
8	1-2	Stop bit	1 Byte
9	0-65535	Send delay	2 Byte
10			
11	3.5-100	Min. Interface	1 Byte
12	!B (8514 Decimal)	Ethernet settings	2 Byte
13	0 /1	use dhcp /static	1 Byte
14	0-255	IP address 1st byte	1 Byte
15	0-255	IP address 2nd byte	1 Byte
16	0-255	IP address 3rd byte	1 Byte
17	0-255	IP address 4th byte	1 Byte

18	0-255	Subnet 1st byte	1 Byte
19	0-255	Subnet 2nd byte	1 Byte
20	0-255	Subnet 3rd byte	1 Byte
21	0-255	Subnet 4th byte	1 Byte
22	0-255	Gateway 1st byte	1 Byte
23	0-255	Gateway 2nd byte	1 Byte
24	0-255	Gateway 3rd byte	1 Byte
25	0-255	Gateway 4th byte	1 Byte
26	2	port no	2 Byte
27			
28	!C(8515 Decimal)	nEthernet settings	2 Byte
29	0 /1	use dhcp /static	1 Byte
30	0-255	IP address 1st byte	1 Byte
31	0-255	IP address 2nd byte	1 Byte
32	0-255	IP address 3rd byte	1 Byte
33	0-255	IP address 4th byte	1 Byte
34	0-255	Subnet 1st byte	1 Byte
35	0-255	Subnet 2nd byte	1 Byte
36	0-255	Subnet 3rd byte	1 Byte
37	0-255	Subnet 4th byte	1 Byte
38	0-255	Gateway 1st byte	1 Byte
39	0-255	Gateway 2nd byte	1 Byte
40	0-255	Gateway 3rd byte	1 Byte
41	0-255	Gateway 4th byte	1 Byte
42	0	port no	2 Byte
43			
44	!D(8516 Decimal)	PLC model	2 Byte
45	1 to 2	XM-14DT/XM-17-ADT	1 Byte
46	!E (8517 Decimal)	Remote IO Configuration	2 Byte
47	0-66/63	No. of Remote DI	1 Byte
48	0-66/63	No. of Remote DO	1 Byte
49	63	No. of Remote AI	1 Byte
50	63	No. of Remote_AO	1 Byte
51		No. of Universal I/P	1 Byte
52		No. of Universal O/P	1 Byte
53	!F (8518 Decimal)	PLC on board IO Configuration	2 Byte
54	8	No. of on-board DI	1 Byte
55	6	No. of on-board DO	1 Byte
56	0/2	No. of on-board AI	1 Byte
57	0/1	No. of on-board AO	1 Byte
58	0 to 3	AI1 mode	1 Byte
59	0 to 3	AI2 mode	1 Byte

60	0 to 3	AO1 mode	1 Byte
61	!G (8519 Decimal)	Modbus Type	2 Byte
62	0-300	No. of Modbus types	2 Byte
63			
64	1 to 3	Modbus_0	1 Byte
65	1 to 3	Modbus_1	1 Byte
66	1 to 3	Modbus_2	1 Byte
67	1 to 3	Modbus_3	1 Byte
68	1 to 3	Modbus_4	1 Byte
69	…	…	…
70	1 to 3	Modbus_299	1 Byte
71	!H (8520 Decimal)	Modbus RTU	2 Byte
72	0-99	No. of RTU request	1 Byte
73	0-255	Slave_ID_0	1 Byte
74	0-255	Slave_ID_1	1 Byte
75	0-255	Slave_ID_2	1 Byte
76	…..	…	…
77	0-255	Slave_ID_98	1 Byte
78	0-255	Slave_ID_99	1 Byte
79	0-3600000	Polling_0	4 Byte
80			
81			
82			
83	0-3600000	Polling_1	4 Byte
84			
85			
86			
87	0-3600000	Polling_2	4 Byte
88			
89			
90			
91	…	…	…
92	0-3600000	Polling_98	4 Byte
93			
94			
95			
96	0-3600000	Polling_99	4 Byte
97			
98			
99			
100	Address in Decimal	Variable_0	4 Byte
101			

102			
103			
104	Address in Decimal	Variable_1	4 Byte
105			
106			
107			
108	Address in Decimal	Variable_2	4 Byte
109			
110			
111			
112	…	…	…
113	Address in Decimal	Variable_98	4 Byte
114			
115			
116			
117	Address in Decimal	Variable_99	4 Byte
118			
119			
120			
121	1-16	Function Code_0	1 Byte
122	1-16	Function Code_1	1 Byte
123	1-16	Function Code_2	1 Byte
124	…	…	….
125	1-16	Function Code_98	1 Byte
126	1-16	Function Code_99	1 Byte
127	0-65535	Address_0	2Byte
128			
129	0-65535	Address_1	2Byte
130			
131	0-65535	Address_2	2 Byte
132			
133	…	…	…
134	0-65535	Address98	2 Byte
135			
136	0-65535	Address_99	2 Byte
137			
138	0-255	Length_0	1 Byte
139	0-255	Length_1	1 Byte
140	0-255	Length_2	1 Byte
141	…	…	…
142	0-255	Length_98	1 Byte
143	0-255	Length_99	1 Byte
144	!I (8521 Decimal)	Modbus TCP Client	2 Byte

145	0-199	No. of TCP client request	1 Byte
146	0-3600000	Polling_0	4 Byte
147			
148			
149			
150	0-3600000	Polling_1	4 Byte
151			
152			
153			
154	0-255	IP address 1st byte_0	1 Byte
155	0-255	IP address 2nd byte_0	1 Byte
156	0-255	IP address 3rd byte_0	1 Byte
157	0-255	IP address 4th byte_0	1 Byte
158	0-255	IP address 1st byte_1	1 Byte
159	0-255	IP address 2nd byte_1	1 Byte
160	0-255	IP address 3rd byte_1	1 Byte
161	0-255	IP address 4th byte_1	1 Byte
162	2-65534	Port_No_0	2 Byte
163			
164	2-65534	Port_No_1	2 Byte
165			
166	0-255	Slave_ID_0	1 Byte
167	0-255	Slave_ID_1	1 Byte
168	Address in Decimal	Variable_0	4 Byte
169			
170			
171			
172	Address in Decimal	Variable_1	4 Byte
173			
174			
175			
176	1-4	Function_Code_0	1 Byte
177	1-4	Function_Code_1	1 Byte
178	0-9998	Address_0	2 Byte
179			
180	0-9998	Address_1	2 Byte
181			
182	0-255	Length_0	1 Byte
183	0-255	Length_1	1 Byte
184	!J (8522 Decimal)	Modbus TCP Server	2 Byte
185	0-199	No. of TCP Server request	2 Byte
186			
187	2-65534	Port no_0	2 Byte

188			
189	2-65534	Port no_1	2 Byte
190			
191	2-65534	Port no_2	2 Byte
192			
193	….	…	…
194	2-65534	Port no_198	2 Byte
195			
196	2-65534	Port no_199	2 Byte
197			
198	Address in Decimal	Variable_0	4 Byte
199			
200			
201			
202	Address in Decimal	Variable_1	4 Byte
203			
204			
205			
206	Address in Decimal	Variable_2	4 Byte
207			
208			
209			
210	…	…	…
211	Address in Decimal	Variable_198	4 Byte
212			
213			
214			
215	Address in Decimal	Variable_199	4 Byte
216			
217			
218			
219	1-4	Function Code_0	1 Byte
220	1-4	Function Code_1	1 Byte
221	1-4	Function Code_2	1 Byte
222	…	…	…
223	1-4	Function Code_198	1 Byte
224	1-4	Function Code_199	1 Byte
225	0-9998	Address_0	2 Byte
226			
227	0-9998	Address_1	2 Byte
228			
229	0-9998	Address_2	2 Byte
230			

231	…	…	…
232	0-9998	Address_198	2 Byte
233			
234	0-9998	Address_199	2 Byte
235			
236	Address in Decimal	Coil variable_0	4 Byte
237			
238			
239			
240	Address in Decimal	Coil variable_1	4 Byte
241			
242			
243			
244	Address in Decimal	Coil variable_2	4 Byte
245			
246			
247			
248	…	…	…
249	Address in Decimal	Coil variable_198	4 Byte
250			
251			
252			
253	Address in Decimal	Coil variable_199	4 Byte
254			
255			
256			
257	Address in Decimal	IP Status Varible_0	4 Byte
258			
259			
260			
261	Address in Decimal	IP Status Varible_1	4 Byte
262			
263			
264			
265	Address in Decimal	IP Status Varible_2	4 Byte
266			
267			
268			
269	…	…	…
270	Address in Decimal	IP Status Varible_198	4 Byte
271			
272			
273			

274	Address in Decimal	IP Status Varible_199	4 Byte
275			
276			
277			
278	Address in Decimal	Holding varibale_0	4 Byte
279			
280			
281			
282	Address in Decimal	Holding varibale_1	4 Byte
283			
284			
285			
286	Address in Decimal	Holding varibale_2	4 Byte
287			
288			
289			
290	...	...	...
291	Address in Decimal	Holding varibale_198	4 Byte
292			
293			
294			
295	Address in Decimal	Holding varibale_199	4 Byte
296			
297			
298			
299	Address in Decimal	Input_variable_0	4 Byte
300			
301			
302			
303	Address in Decimal	Input_variable_1	4 Byte
304			
305			
306			
307	Address in Decimal	Input_variable_2	4 Byte
308			
309			
310			
311	...	...	...
312	Address in Decimal	Input_variable_198	4 Byte
313			
314			
315			
316		Input_variable_199	4 Byte

317	Address in Decimal		
318			
319			
320	0-255	Length_0	1 Byte
321	0-255	Length_1	1 Byte
322	0-255	Length_2	1 Byte
323	…	…	…
324	0-255	Length_198	1 Byte
325	0-255	Length_199	1 Byte
326	!K (8523 Decimal)	Retentive	2 Byte
327	0-300	No. Of retentive address	2Byte
328			
329	Address in Decimal	Retain add_0	4 Byte
330			
331			
332			
333	Address in Decimal	Retain add_1	4 Byte
334			
335			
336			
337	Address in Decimal	Retain add_2	4 Byte
338			
339			
340			
341	…	…	…
342	Address in Decimal	Retain add_298	4 Byte
343			
344			
345			
346	Address in Decimal	Retain add_299	4 Byte
347			
348			
349			
350	Address in Decimal	Logical add_0	4 Byte
351			
352			
353			
354	Address in Decimal	Logical add_1	4 Byte
355			
356			
357			
358		Logical add_2	4 Byte

359	Address in Decimal		
360			
361			
362	...	...	...
363	Address in Decimal	Logical add_298	4 Byte
364			
365			
366			
367	Address in Decimal	Logical add_299	4 Byte
368			
369			
370			
371	!L (8524 Decimal)	Initial	2 Byte
372	0-600	No. Of Initial address	2Byte
373			
374	Address in Decimal	Logical add_0	4 Byte
375			
376			
377			
378	Address in Decimal	Logical add_1	4 Byte
379			
380			
381			
382	Address in Decimal	Logical add_2	4 Byte
383			
384			
385			
386	Address in Decimal	Logical add_3	4 Byte
387			
388			
389			
390	...	...	...
391	Address in Decimal	Logical add_298	4 Byte
392			
393			
394			
395	Address in Decimal	Logical add_299	4 Byte
396			
397			
398			
399	Address in Decimal	Init val_0	4 Byte
400			
401			

402			
403	Address in Decimal	Init val_1	4 Byte
404			
405			
406			
407	Address in Decimal	Init val_2	4 Byte
408			
409			
410			
411	Address in Decimal	Init val_3	4 Byte
412			
413			
414			
415	...	...	...
416	Address in Decimal	Init val_598	4 Byte
417			
418			
419			
420	Address in Decimal	Init val_599	4 Byte
421			
422			
423			
424	!M (8525 Decimal)	Expansion	2 Byte
425	0-5	No. of Expansion	1 Byte
426	6	XM-DI-Code	2 byte
427			
428	Address in Decimal	Logical bit add_0	4 Byte
429			
430			
431			
432	Address in Decimal	Logical bit add_1	4 Byte
433			
434			
435			
436	Address in Decimal	Logical bit add_2	4 Byte
437			
438			
439			
440	Address in Decimal	Logical bit add_3	4 Byte
441			
442			
443			

444	Address in Decimal	Logical bit add_4	4 Byte
445			
446			
447			
448	Address in Decimal	Logical bit add_5	4 Byte
449			
450			
451			
452	Address in Decimal	Logical bit add_6	4 Byte
453			
454			
455			
456	Address in Decimal	Logical bit add_7	4 Byte
457			
458			
459			
460	Address in Decimal	Logical bit add_8	4 Byte
461			
462			
463			
464	Address in Decimal	Logical bit add_9	4 Byte
465			
466			
467			
468	Address in Decimal	Logical bit add_10	4 Byte
469			
470			
471			
472	Address in Decimal	Logical bit add_11	4 Byte
473			
474			
475			
476	Address in Decimal	Logical bit add_12	4 Byte
477			
478			
479			
480	Address in Decimal	Logical bit add_13	4 Byte
481			
482			
483			
484	Address in Decimal	Logical bit add_14	4 Byte
485			
486			

487			
488	Address in Decimal	Logical bit add_15	4 Byte
489			
490			
491			
492	10	XM-D0-Code	2 byte
493			
494	Address in Decimal	Logical bit add_0	4 Byte
495			
496			
497			
498	Address in Decimal	Logical bit add_1	4 Byte
499			
500			
501			
502	Address in Decimal	Logical bit add_2	4 Byte
503			
504			
505			
506	Address in Decimal	Logical bit add_3	4 Byte
507			
508			
509			
510	Address in Decimal	Logical bit add_4	4 Byte
511			
512			
513			
514	Address in Decimal	Logical bit add_5	4 Byte
515			
516			
517			
518	Address in Decimal	Logical bit add_6	4 Byte
519			
520			
521			
522	Address in Decimal	Logical bit add_7	4 Byte
523			
524			
525			
526	Address in Decimal	Logical bit add_8	4 Byte
527			
528			
529			

530	Address in Decimal	Logical bit add_9	4 Byte
531			
532			
533			
534	Address in Decimal	Logical bit add_10	4 Byte
535			
536			
537			
538	Address in Decimal	Logical bit add_11	4 Byte
539			
540			
541			
542	Address in Decimal	Logical bit add_12	4 Byte
543			
544			
545			
546	Address in Decimal	Logical bit add_13	4 Byte
547			
548			
549			
550	Address in Decimal	Logical bit add_14	4 Byte
551			
552			
553			
554	Address in Decimal	Logical bit add_15	4 Byte
555			
556			
557			
558	12	XM-DI8_DO6-Code	2 byte
559			
560	Address in Decimal	Logical bit add_0	4 Byte
561			
562			
563			
564	Address in Decimal	Logical bit add_1	4 Byte
565			
566			
567			
568	Address in Decimal	Logical bit add_2	4 Byte
569			
570			
571			
572		Logical bit add_3	4 Byte

573	Address in Decimal		
574			
575			
576	Address in Decimal	Logical bit add_4	4 Byte
577			
578			
579			
580	Address in Decimal	Logical bit add_5	4 Byte
581			
582			
583			
584	Address in Decimal	Logical bit add_6	4 Byte
585			
586			
587			
588	Address in Decimal	Logical bit add_7	4 Byte
589			
590			
591			
592	Address in Decimal	Logical bit add_0	4 Byte
593			
594			
595			
596	Address in Decimal	Logical bit add_1	4 Byte
597			
598			
599			
600	Address in Decimal	Logical bit add_2	4 Byte
601			
602			
603			
604	Address in Decimal	Logical bit add_3	4 Byte
605			
606			
607			
608	Address in Decimal	Logical bit add_4	4 Byte
609			
610			
611			
612	Address in Decimal	Logical bit add_5	4 Byte
613			
614			
615			

616	13	XM-AI2_AO2-Code	2 byte
617			
618	Address in Decimal	Logical bit add_0	4 Byte
619			
620			
621			
622	Address in Decimal	Logical bit add_1	4 Byte
623			
624			
625			
626	Address in Decimal	Logical bit add_0	4 Byte
627			
628			
629			
630	Address in Decimal	Logical bit add_1	4 Byte
631			
632			
633			
634	1-3	AIAO_Mode_0	1 Byte
635	1-3	AIAO_Mode_1	1 Byte
636	1-3	AIAO_Mode_2	1 Byte
637	…	…	…
638	1-3	AIAO_Mode_24	1 Byte
639	!N (8526 Decimal)	Universal I/O	2 Byte
640	5	No. of universal I/O	1 Byte
641	1-10	UIUO Mode_0	1 Byte
642	1-10	UIUO Mode_1	1 Byte
643	1-10	UIUO Mode_2	1 Byte
644	1-10	UIUO Mode_3	1 Byte
645	1-10	UIUO Mode_4	1 Byte
646	1-10	UIUO Mode_5	1 Byte
647	…	…	…
648	1-10	UIUO Mode_29	1 Byte
649	1-10	UIUO Mode_30	1 Byte
650	Address in Decimal	UIUO Address_0	4 Byte
651			
652			
653			
654	Address in Decimal	UIUO Address_1	4 Byte
655			
656			
657			

658	Address in Decimal	UIUO Address_2	4 Byte
659			
660			
661			
662	Address in Decimal	UIUO Address_3	4 Byte
663			
664			
665			
666	Address in Decimal	UIUO Address_4	4 Byte
667			
668			
669			
670	Address in Decimal	UIUO Address_5	4 Byte
671			
672			
673			
674	…	…	…
675	Address in Decimal	UIUO Address_59	4 Byte
676			
677			
678			
679	Address in Decimal	UIUO Address_60	4 Byte
680			
681			
682			
683	!NO(8527 Decimal)	CRC	2 Byte
684		CRC	2 Byte
685			
686	&	EOF	1 Byte

- Phase 1-We will complete this task and we need both App.csv, Config.csv along with MCode and CCode.
- Also generate MCode and CCode text (.bin) files in project. (For checking the correct conversion)
- We will remove both .csv files once we complete Phase1. (Messung will decide)
- After implementation of MCode and CCode in XMPS-2000 compiler may take more time to compile the project, Add one pop-up window to show compile status.(no time limitation for compile process)
- For user entered address conversion logic (for MCode, CCode) please refer SRS_Ref_code.txt file.

Download MCODE & CCODE into hardware via TFTP.

- For now to download MCode, CCode we will use any unused button from XMPS-2000 window (like find, next screen, prev. screen)
- Later after testing and proper working we will shoft to download button.

Some minor changes will happen will update this when we reach to this stage. (Addition of extra bytes and extra settings etc.)

3. Retentive timer instruction block

Retentive timer instruction block is same as Timer TON (0.01s, 0.1s, 1s), Retentive timer which retain accumulated value, In case of power loss, PLC restart.

Timer starts from last state of Elapsed time, Elapsed time value is maintained by this timer in case if INPUT Bit goes off, power loss, PLC restart.

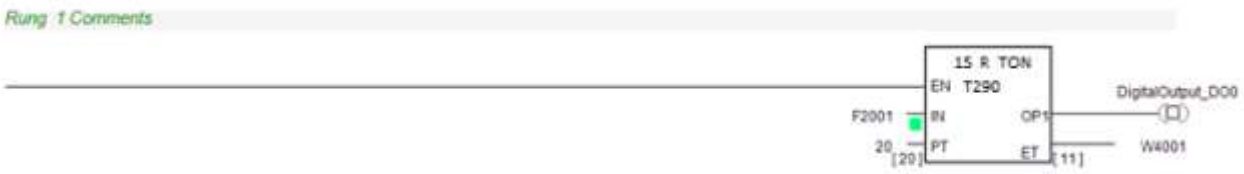
Below table describes timer name to add in app.csv file (T/C Name)

Timer Type	Timer Name
0.01s RTON Timer	T256 to T272
0.1s RTON Timer	T273 to T289
1s RTON Timer	T290 to T306

For Retentive timer TON new OPCODE will be add.

Timer Type	Function	OPCODE	Data Type
Retentive Timer TON	0.01s	0X034B	RTON
	0.1s	0X035B	
	1s	0X036B	

- In given OPCODE instruction types are
  1. 0.01s RTON - 0X034X
  2. 0.1s RTON - 0X035X
  3. 1s RTON - 0X036X
- And Data type is 0X000B for all 3 functions of RTON
- Block name will be 1s RTON
- Example1 :



In above example after power restart or IN bit reset (off then on) Retentive Timer will resume Elapsed timer count from 11 to 20 and then OP1 (Done bit) will turn ON.

App.csv for above example–

Line Numb	T/C Name	OutputTyp	DataType	Enable	Output1	Output2	Op Code	Input1	Input2	Input3	Input4	Comments	WindowName
1	T290	0	B	-	Q0:000.00	W4:001	036B	F2:001	20	-	-		B01

4. Generate Disabling variables for Modbus RTU request

Add Disabling variable to the Modbus RTU master request.

1. One column will be going to add to the right side of Tag with name Disabling variables.
2. When user right click on Modbus RTU request, select second option Generate Disabling Variable.
3. After selecting this option Disabling variable column get enable for that request and show dropdown arrow, then user can add new tag or can select existing tag from list (Only bit address).
4. For Generate disabling variable only physical bit addresses and memory bit addresses should validate.
5. If user add Disabling address, selected address should be added in Disabling variables column.
6. If user doesn't add any disabling address, then 0 should be add in .csv file.

Modbus RTU Master request table,

Polling	Device ID	Address	Length	Variable	Function Code	Tag	Disabling Variables
100	1	0	1	W4:000	Read Holding Registers (03)	W400	F2:000
200	2	1	1	W4:001	Read Holding Registers (03)	W4001	I1:000.01

Config.csv file

ConfigTyp	Alternate IO	ModbusTy	Slave ID	Communic	NoOfRetri	Polling	SlaveIPAdd	TCPPort	Variable	Data Start	Data Size	FunctionCo	IO List	Model	Type	Mode	Label	Logical Address	Tag	RetAdd
IO_Mapping																-	DI3	I1:000.03	DigitalInput_DI3	
IO_Mapping																-	DI4	I1:000.04	DigitalInput_DI4	
IO_Mapping																-	DI5	I1:000.05	DigitalInput_DI5	
IO_Mapping																-	DI6	I1:000.06	DigitalInput_DI6	
IO_Mapping																-	DI7	I1:000.07	DigitalInput_DI7	
IO_Mapping															DigitalOutput	-	DO0	Q0:000.00	DigitalOutput_DO0	
IO_Mapping																-	DO1	Q0:000.01	DigitalOutput_DO1	
IO_Mapping																-	DO2	Q0:000.02	DigitalOutput_DO2	
IO_Mapping																-	DO3	Q0:000.03	DigitalOutput_DO3	
IO_Mapping																-	DO4	Q0:000.04	DigitalOutput_DO4	
IO_Mapping																-	DO5	Q0:000.05	DigitalOutput_DO5	
Modbus			1	1		100			W4:000	0	1	3						F2:000		
Modbus			1	2		200			W4:001	1	1	3						I1:000.01		

5. Curser Tool-tip feature in online & offline mode

A. For contacts and coils the format should be in ITALIC as-

ADDRESS : BOOL : INITIAL VALUE (if any else blank) : RETAIN (if retained else blank);

Example -

F2:000 : BOOL : 1 : RET;

I1:000.00 : BOOL : : ;

If the cursor is placed on the variable, wait for 2 secs then display the information for 3 secs.

B. For FBs the format should be in ITALIC as-

(ADDRESS : DATA TYPE : INITIAL VALUE (if any else blank) : RETAIN (if retained else blank);)

Example of ADD -

(IN1 : P5:000 : REAL : 100.55 : RET;

IN2 : P5:001 : REAL : : ;

IN3 : P5:002 : REAL : 80.17 : RET;

IN4 : P5:003 : REAL : : ;

OP1 : P5:099 : REAL : : ;)

If the cursor is placed on the FB, wait for 2 secs then display the information for 5 secs.

Note : Not used Inputs should not display in tool tip.