本文档使用原始的 INS 指令集作为演示,您可以导出 COP2000 的原始指令集或者直接运行一次 ins2table.py 文件即可生成原始指令集。

接下来对 INS 文件结构进行讲解:

下图是 INS 文件的首部 header,请不要修改。

@	00	01	02	03	04	05	06	07	08	09	ØA.	0B	øс	ØD.	ØE	0F	D	ec	ode	ed T	ext								
00000000			53	54	55	52	43	54							75	4D								0			&		М
00000010	20			6C	65				72	20	43			32															
00000020	30	1 A	01	07			41	54	43			00	00	00	46	CA		ø											
00000030	В5				FA	D5	вс		С3	А3	AC	В2	BB		C9	DØ		μ				õ	Ó						
00000040	DE	B8	C4	A1	А3	B8	В4	CE	ВВ	ВА		АЗ		СВ	F9														Ó
00000050	DØ	вс	C4	B4	E6	C6	F7	C 7	E5		АЗ	AC	CA	D7	CF	C8													
00000060	D6	B4	DØ	DØ			46	41	54	43			20	D6	В8	C1		ö										Ö	
00000070	EE	C8	A1.	D6	В8	00	00	00	01	03	41	44	44	10	01	02					Ö				Α	D	D		

接下来, 0x00 开头表示是一条空指令, 共三条。

4	Α	В	С	D	E	F	G
1		mnemoni		address			
2	1	_FATCH_	CBFFFF00	FFFFF00	FFFFF00	FFFFF00	0x00
3	2						0x04
4	3						0x08
5	4						0x0C
6	5	ADD A,R?	FFF7EFFF	FFFE90FF	CBFFFFFF	FFFFFF00	0x10
7	6	ADD A,@I	FF77FFFF	D7BFEFFF	FFFE90FF	CBFFFFF	0x14

0x01 开头代表此处有指令, 0x03 代表指令名占接下来的 3 个 byte.指令名之后的 0x10 标明 了指令地址, 0x01 代表第一个操作数是 A 寄存器, 0x02 代表第二个操作数是 R?, 0x00 是指令结束标志



最后一条 RETI 指令也一样。0x01 表示有指令,04 表示指令名占接下来的 4 个 byte.指令名之后的 0xEC 标明了指令地址,0x00 代表第一个操作数空,0x00 代表第二个操作数空,0x00 是指令结束标志。后面四条 0x00 标明接下来有四条空指令。



_INT_指令结构特殊,内容和位置都不允许修改。

000001C0	05	00	00	00	01	03	4A	4D	50	AC	05	00	00	00	00	01		4)	4		100	i.	J	М	Р	7	0.85			44	
000001D0	05				54		В8	00	00		CA	В5	D1	E9	ВВ	FA															
000001E0	D5	ВС	D3	С3	А3	AC	В2	BB		C9	D0	DE	В8	C4	A1	А3	õ		Ó												
000001F0	BD	F8	C8	ЕВ	D6	D0	В6	CF	CA	В1	АЗ	AC	CA	B5	D1	E9					Ö		9								
00000200		FA	D3	В2	BC	FE	В2	FA	C9	FA	20				54				Ó												
00000210	20	D6	В8	C1	EE	01	04	43	41	4C	4C	ВС	05	00	00	01		Ö		Á	î	٠		С	Α	L	L	1/4	٠	4	

指令描述结束后,是文件的微指令存储区。每条微指令有 4 个 byte, 每条指令有四条微指令, 共 64 条指令、256 条微指令。在 INS 文件中,每条微指令都需要 8 位 hex 表示。最后两位要么是 0x00,要么是 0xff。正常来说,每条微指令的最后两位都是以 0xff 结尾,空微指令 0xffffff 才以 0x00 结尾。不过也有一些例外,比如 READ、WRITE、_INT_等指令的微指令可能会以 0x00 结尾。具体原因未知,不过不建议改动这几条特殊指令。微指令按字节逆序存储并补全末尾一字节,例如一条微指令 0xD7 0xEF 0xBF 会被存储为 0xBF 0xEF 0xD7 0xFF。