RoboNova and Robobasic Intermediate Byte Code

Revision 0.1

23th July 2006

Introduction

This information on the Intermediate Byte code used by the is collected from the tokenised codes downloaded from RoboBasic to the RoboNova and subsequently executed.

The RoboBASIC development environment.

The development environment comprises the following components:

- Program Editor
- Capability to perform Direct commands
- Real time control
- Downloader
- Intermediate Code Builder

The Downloader is described in a separate document which shows the PC to RoboNova MR-C3024 serial command set.
http://web.ukonline.co.uk/r.ibbotson/files/C3024Serial.pdf
The downloader transfers the program header and the tokenised intermediate code to the External EEPROM memory on the MR-C3024. The download dialog starts at 9600 bps and shifts speed to 115K for the actual data transfer.
The downloaded files are entirely self contained, and could be saved (not in Robobasic) and downloaded by a different application.

The Intermediate code builder:

- Tokenizes the Program from the editor or .rsf file
- Allocates variables to actual Data memory locations
- Inserts actual Program EEPROM memory locations into IM Code
- Replaces some Keywords with multiple tokens

The RoboScript development environment.

The RoboScript development environment is a cut down and simplified version of Robobasic. It generates identical intermediate code, but not as rich.

How to view the IM Code

The .bas and .rsf files stored by RoboBasic and Roboscript are in source file format and not in IM code. The converted or compiled code is not stored.
The easiest way to see the code is to install a serial port monitor on the COM port used for the interface. I use the freeware:
http://www.serial-port-monitor.com/
Find the COM port used by the serial interface to the C3024, and install the monitor software on this port, before starting RoboBasic or RoboScript.

After starting Robobasic, the serial port monitor will capture all the PC to C3024 commands as described in:
http://web.ukonline.co.uk/r.ibbotson/files/C3024Serial.pdf

When the download command is invoked, then you can see the IM code.

Intermediate Code Storage in MR-C3024

The intermediate code is stored by the downloader into the external EEPROM of the controller.

The first 16 bytes of the EEPROM contain the program header:

Byte	Usage
0 - 7	Program Name
8	Year
9	Month
10	Day
11	Hour
12	Minute
13	Seconds
14	Length in Bytes Low
15	Length in Bytes High

The tokenised code follows and starts at location 16 (0x10)

Intermediate Code Tokens

The following shows the tokens identified so far:

Token Value Decimal	Token Value HEX	Token Type	Usage	Notes
0	00	NULL		
1	01			
2	02			
3	03			
4	04			

5	05			
6	06			
7	07			
8	08			
9	09			
10	0A			
11	0B			
12	0C			
13	0D			
14	0E			
15	0F			
16	10	Literal 0		
17	11	Literal 1		
18	12	8 bit Literal	1 byte number value follows	
19	13	16 bit Literal	2 byte number value follows	
20	14			
21	15	Byte Variable	1 byte number address follows	
22	16	Integer Variable	1 byte number address follows	
23	17			
24	18			
25	19			
26	1A			
27	1B	"." BIT	1 byte number address, then 1 byte number bitmap follows	for bits in integers address 2nd byte as byte at address + 1
28	1C			
29	1D			
30	1E			
31	1F			
32	20			
33	21	Arithmetic +		
34	22	Arithmetic -		
35	23	Arithmetic *		
36	24	Arithmetic /		
37	25			
38	26	Arithmetic %		
39	27	Logic AND		
40	28	Logic OR		
41	29			
42	2A	Logic XOR		
43	2B			
44	2C	Logic >>		
45	2D	Logic <<		
46	2E			
47	2F			
48	30	Relational <		
49	31	Relational >		

50	32	Relational <=		
51	33	Relational >=		
52	34	Relational <>		
53	35	Relational =		
54	36			
55	37			
56	38			
57	39			
58	3A			
59	3B			
60	3C			
61	3D			
62	3E			
63	3F			
64	40			
65	41			
66	42			
67	43			
68	44			
69	45			
70	46			
71	47			
72	48			
73	49			
74	4A			
75	4B			
76	4C			
77	4D			
78	4E			
79	4F			
80	50			
81	51			
82	52			
83	53			
84	54			
85	55			
86	56			
87	57			
88	58			
89	59			
90	5A			
91	5B			
92	5C			
93	5D			
94	5E			
95	5F			
96	60			
97	61			
98	62			
99	63			
100	64			
101	65			

102	66			
103	67			
104	68			
105	69			
106	6A			
107	6B			
108	6C			
109	6D			
110	6E			
111	6F			
112	70			
113	71			
114	72			
115	73			
116	74			
117	75			
118	76			
119	77			
120	78			
121	79			
122	7A			
123	7B			
124	7C			
125	7D			
126	7E			
127	7F			
128	80			
129	81			
130	82			
131	83			
132	84			
133	85			
134	86			
135	87			
136	88			
137	89			
138	8A			
139	8B			
140	8C			
141	8D			
142	8E			
143	8F			
144	90			
145	91			
146	92			
147	93			
148	94			
149	95			
150	96			
151	97			
152	98			
153	99			

154	9A			
155	9B			
156	9C			
157	9D			
158	9E			
159	9F			
160	A0			
161	A1			
162	A2			
163	A3			
164	A4			
165	A5			
166	A6			
167	A7			
168	A8			
169	A9			
170	AA			
171	AB			
172	AC			
173	AD			
174	AE			
175	AF			
176	B0	MOVE		
177	B1			
178	B2			
179	B3	ZERO		
180	B4	DIR		
181	B5			
182	B6	PTP ALL, HIGHSPEED	00 = high speed off, 01 = highspeed on, 05 = PTP ALLOFF, 06 = PTP ALLON	
183	B7			
184	B8			
185	B9			
186	BA	MUSIC		
187	BB			
188	BC			
189	BD			
190	BE			
191	BF	TEMPO		
192	C0	IF		
193	C1	FOR		
194	C2	NEXT		
195	C3	TO		
196	C4	GOTO		
197	C5	GOSUB		
198	C6	RET		
199	C7			
200	C8			
201	C9			
202	CA			

203	CB			
204	CC	POKE		
205	CD	ROMPOKE		
206	CE			
207	CF			
208	D0	Assign	lvalue, expression	expressions are enumerated from left to right, no priority
209	D1	OUT		
210	D2	PULSE		
211	D3	TOGGLE		
212	D4	DELAY	value expression follows follows	
213	D5			
214	D6			
215	D7			
216	D8			
217	D9			
218	DA	MOTOR	literal	Motor G24, G8x, G6x, are all made to individual motor commands
219	DB	MOTOROFF		
220	DC			
221	DD	SPEED		
222	DE	PWM		
223	DF	SERVO		
224	E0	LCDINIT		
225	E1	CLS		
226	E2	LOCATE		
227	E3	PRINT		
228	E4			
229	E5			
230	E6			
231	E7	BYTEOUT		
232	E8			
233	E9	WAIT		
234	EA	STOP		
235	EB	RUN		
236	EC			
237	ED	PTP SET	00 = off, 01 = on	
238	EE			
239	EF	NOT		NOT()
240	F0	IN		
241	F1	KEYIN		
242	F2	BYTEIN		
243	F3			
244	F4			
245	F5			
246	F6	STATE		
247	F7			

248	F8	RND		
249	F9	PEEK		
250	FA	ROMPEEK		
251	FB			
252	FC			
253	FD			
254	FE			
255	FF	? NOP		

Variables

Variables are allocated by the RoboBASIC tokeniser into the data memory region addressed by a single byte address. Only bytes (8 bits) or Integers(16bits) are allowed. Varables start at location 0x40 (actually 0x140 in ATMega address space) , and are available up to location 0xFE.

Robobasic Commands

The following describes how Robobasic commands are translated to tokens:

Declaration/Definition

The commands DIM, AS, CONST, BYTE, INTEGER, do not create tokens. They are used to create storage locations as either a byte or integer which are sequentially allocated in data memory starting at location 0x140. Subsequent referenences to these locations is made using the “Byte Variable” (0x15) or “Integer Variable” (0x16) tokens, where the token is followed by a single byte giving the data memory address.

Flow Control Commands

The commands “IF, THEN, ELSE, ELSEIF, ENDIF” are reduced to a sequence of simple conditional jump instructions by RoboBASIC. These are of the form:

Byte 1	IF Token (0xC0)
Byte 2	Low order jump address if condition not satisfied
Byte 3	High order jump address if condition not satisfied
Byte 4..	NULL terminated condition statement

For example:

```
10 DIM a as byte
20 DIM b as byte
30 IF a = 0 THEN b = 0
40 ELSEIF a = 1 THEN b = 1
50 ELSE b = 2
60
```

Translates to:

C0 = If Token
= Physical EEPROM address of Line 40

= Null terminated condition statement (a=0 ?)
= (b = 0)

C0 =if Token
= Physical EEPROM address of Line 50
= Null terminated condition statement (a=1 ?)
= (b = 1)
= (b = 2)

The commands FOR, TO, NEXT are translated to 3 separate tokens with the following format:

FOR (0xC1), variable token, literal token(start value)
TO (0xC3), variable token, literal token (end value), physical EEPROM location after for loop
NEXT (0xC2), variable token, physical EEPROM address of TO instruction.

The commands GOTO, GOSUB, and RETURN are translated to 3 different tokens with the following format:

GOTO (0xC1), physical EEPROM location of jump destination

GOSUB (0xC3), physical EEPROM location of subroutine

RETURN (0xC2)