SECTION 6. PROGRAMMING CONSIDERATIONS

PC-700/900 Programmable Controllers

6-1. MEMORY AND REGISTER USE

When memory has been erased from a **Numa-Logic** Programmable Controller, the program loader displays the number of words remaining. This number is always two less than anticipated (e.g., 8192 is displayed as 8190, 1536 is shown as 1534, etc.). This occurs because two words of the total memory are not available for use, and, therefore, are not included in the "words available" tabulation.

The reason that these two words of memory are unavailable is that one is used to designate the End of Program (EOP) and the other indicates the first holding register (HR0001). (See Figure 6-1.) The program, consisting of ladder diagrams, uses memory from the bottom, pushing up the location of the EOP word after checking to ensure that memory is available for each ladder diagram before it is input.

Registers are assigned memory from the top (as illustrated in Figure 6-1), starting at HR0001. The **Numa-Logic** Programmable Controller assumes that a reference has been made to HR0001, even if no such reference has been made. The controller keeps track of the Highest Register Reference Used (HRRU) and uses this value (in reference to HR0001) to set aside a block of (user) memory for holding register use. If, for example, HR0200 is the only holding register reference made in the program, 200 words of user memory are set aside even though no other holding registers are used. When programming, it is important to reference holding registers starting with HR0001.

Noting the significance of the HRRU, the amount of memory required for a given program can be easily established by using Table 6-1. For example, a simple program might consist of the following elements:

10 - IN contacts	@ 1 word	=	10 words
3 - CR contacts	@ 1 word	=	3 words
2 - Timers	@ 3 words	=	6 words
1 - Down Counter	@ 3 words	=	3 words
3 - CR coils	@ 1 word	=	3 words
			25 words
	HRRU =		6 words
			31 words
	EOP =		1 word
			32 words total

6-2. PROM-EQUIPPED UNITS

The **Numa-Logic** PC-900 is available with a mix of Programmable Read Only Memory (PROM) and Random Access Memory (RAM) in three versions. These are illustrated in Figure 6-2. Several requirements and variable conditions must be considered before the appropriate mix can be selected. Consider the following:

- The ladder diagram must be in PROM. (For this reason, the PROM-equipped units are not available in an online programming version.)
- Seven words of PROM must be allocated past the EOP for internal "housekeeping" functions. The memory remaining must also be greater than or equal to seven.
- RAM is devoted to holding registers (HRs). Thus, with 512 words of RAM, the first holding register locations in PROM would be HR0513, and with 1536 words of RAM, HR1537.
- Battery backup may be used to retain data in RAM. If, however, the battery backup is not used, RAM is zeroed at power up, setting the output registers and holding registers (located in RAM) to zero, as well as clearing force conditions.
- Contents that must be periodically changed (i.e., fine-tuned) must be put into RAM.

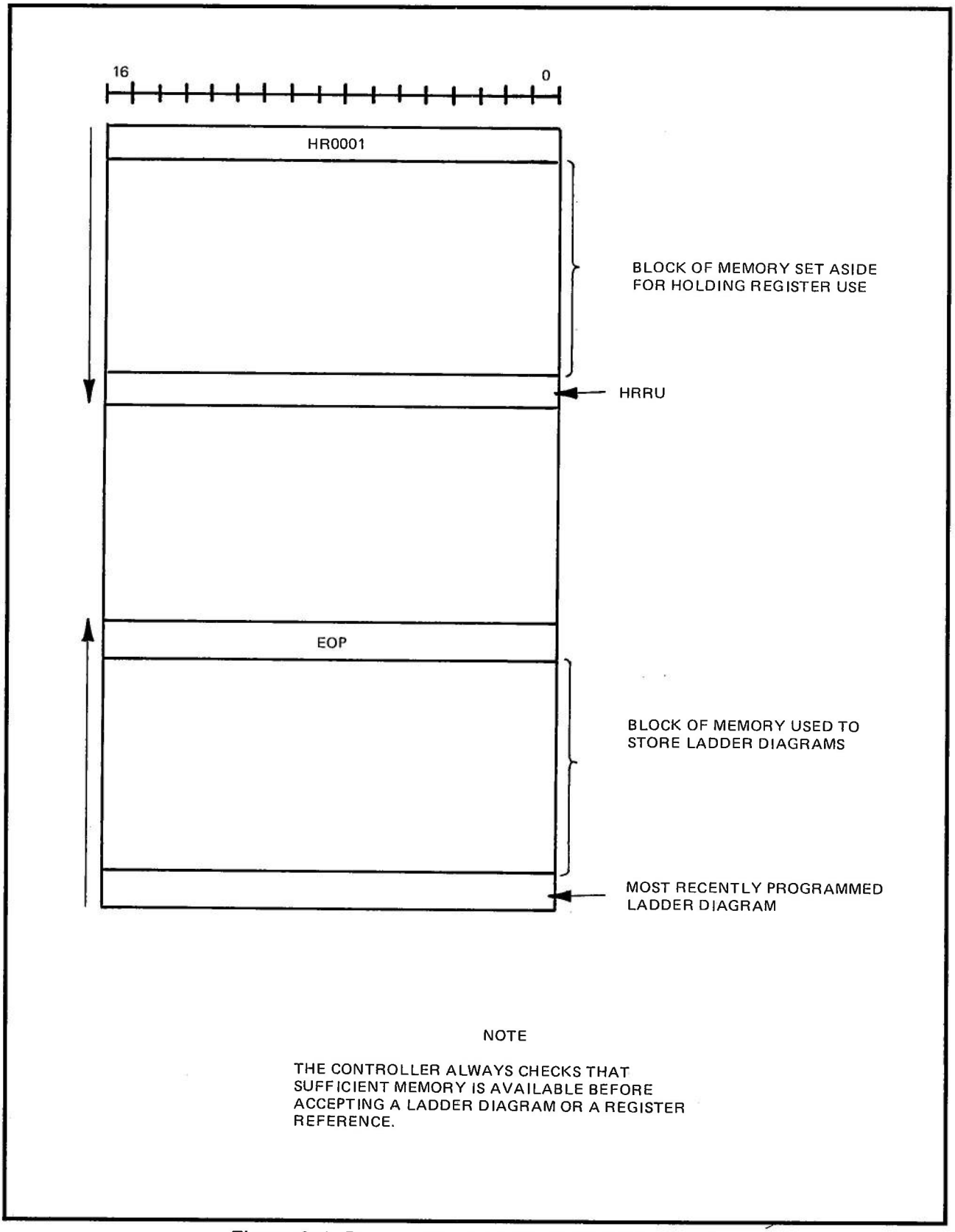

Figure 6-1. Program/Register Memory Assignment

TABLE 6-1. MEMORY/REGISTER USE

Function	Mnemonic	Memory Use (Words)	Input Designation	Data Storage and Register Options					
				Constant Value	HR	IR	OR	IG	OG
Add, Subtract	AD,SB	4	Operand 1		•	•	•		
			Operand 2	•	•	•	•		
			Destination		•		•		
AND Matrix, OR Matrix XOR Matrix	AM,OM, XM	6	Matrix Size	•					
			Matrix 1 End		•	•	•	•	•
			Matrix 2 End		•	•	•	•	•
			Destination End		•		•		•
Ascending Sort	AS	5	Table Length	•					
			Table 1 End		•				
			Table 2 End		•				
ASCII Transmit	AT	6	Return Register		•		•		•
			Table End		•				
			Pointer		•		•		•
			Destination	•					
Bit Operate	BO	5	Table Length	•					
			Table End		•	•	•	•	•
			Pointer		•	•	•	•	•
Bit Pick Contact, NO or NC	BP	2			•	•	•		
Bit Set, Bit Clear, Bit Follow	BS,BC, BF	3	Bit Number	•					
			Register Type		•		•		
Block Transfer	BT	5	Table Length	•					
			Source End		•	•	•	•	•
			Destination End		•		•		•
Close Table	CT	6	Table Length	•					
			Table End		•				
			Pointer		•	•	•	•	•
			Destination		•	•	•	•	•

TABLE 6-1. MEMORY/REGISTER USE (Cont'd)

Function	Mnemonic	Memory Use (Words)	Input Designation	Data Storage and Register Options					
				Constant Value	HR	IR	OR	IG	OG
Comparisons	EQ,GE	3	Operand 1		•	•	•		
			Operand 2	•	•	•	•		
Complement Matrix	CM	5	Matrix Size	•					
			Source End		•	•	•	•	•
			Destination End		•		•		•
Continuous Group Select	CG	3	Operand 1	•					•
Continuous Select	CS	3	Operand 1	•					
Conversions, Move	BD,DB, MV	4	Source		•	•	•	•	•
			Destination		•		•		•
Control Relay Contact, NO or NC	CR	1							
Counters	UC,DC	3	Preset	•	•	•	•		
			Actual		•		•		
Divide	DV	4	Operand 1 (Pair)		•	•	•		
			Operand 2	•	•	•	•		
			Destination (Pair)		•		•		
Drum Controller	DR	5	Number of Steps	•					
			Starting Register		•				
			Step Pointer		•		•		
			Destination				•		•
End of Program	EOP	1							
First In Stack Register-to-Table Move	FI RT	6	Table Length	•					
			Table End		•		•		•
			Pointer		•		•		
			Source		•	•	•	•	•
First Out Fetch Last Out Fetch	FO LO	6	Table Length	•					
			Table End		•		•		•
			Pointer		•		•		
			Destination		•		•		•
Input Contact, NO or NC	IN	1							

TABLE 6-1. MEMORY/REGISTER USE (Cont'd)

Function	Mnemonic	Memory Use (Words)	Input Designation	Data Storage and Register Options					
				Constant Value	HR	IR	OR	IG	OG
I/O Update Immediate	UI	2	I/O Register Group/Pair			•		•	
Loop Controller	LC	6	Set Point	•	•	•	•	•	•
			Process Variable		•	•	•	•	•
			Output		•		•		•
			Loop Table End		•				
Master Control Relay, Skip	MR,SK	2	Number of Coils	•					
Multiply	MP	4	Operand 1		•	•	•		
			Operand 2	•	•	•	•		
			Destination (Pair)		•		•		
N Bit Serial Shift Register	NR,NL	5	Table Length	•					
			Table End		•		•		
			N Bit Field	•	•	•	•		
Open Table	OT	6	Table Length	•					
			Table End		•				
			Pointer		•	•	•	•	•
			Source		•	•	•	•	•
Reset Watchdog Timer	RW	3	Operand 1		•				
Restore Program Counter	RP	3	Operand 1		•				
Save Program Counter	SP	3	Operand 1		•				
Search Matrix	SM	5	Matrix Size	•					
			Matrix End		•	•	•	•	•
			Bit Register		•		•		•
Shift Registers	SR,SL	3	Starting Register		•		•		
			Number of Registers	•					
Square Root	SQ	4	Source		•	•	•		
			Destination		•		•		

TABLE 6-1. MEMORY/REGISTER USE (Cont'd)

Function	Mnemonic	Memory Use (Words)	Input Designation	Data Storage and Register Options					
				Constant Value	HR	IR	OR	IG	OG
Table Lookup Table Lookup Ordered	TL TO	6	Table Length	•					
			Table End		•				
			Pointer		•		•		•
			Source		•	•	•	•	•
Table-to-Register Move	TR	6	Table Length	•					
			Table End		•	•	•	•	•
			Pointer		•		•		
			Destination		•		•		•
Timers	TS,TT	3	Preset	•	•	•	•		
			Actual		•		•		
Update Select	US	4	I/O Quarter	•					
			Dummy Operand	•					

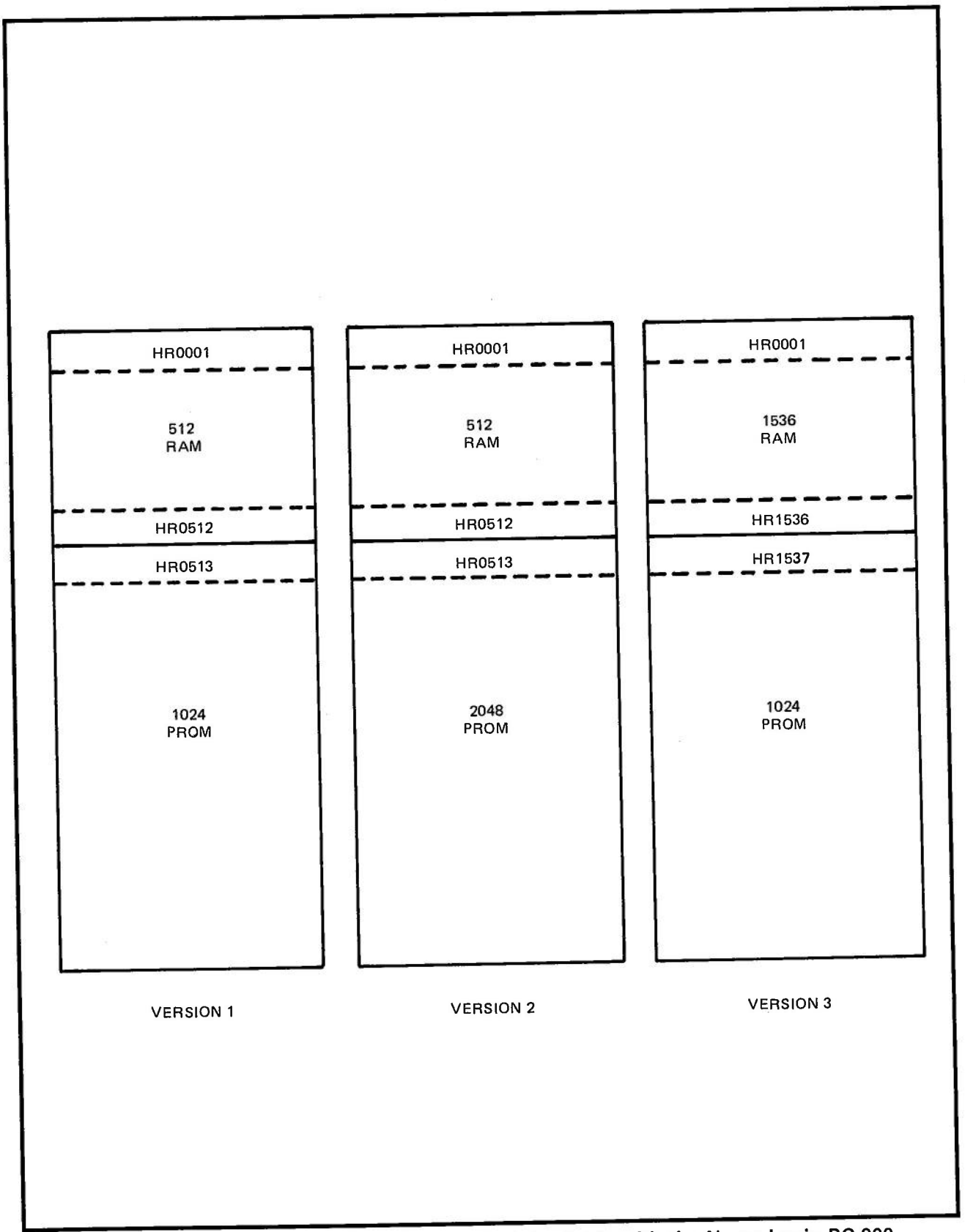

Figure 6-2. Combinations of RAM and PROM Available with the Numa-Logic PC-900

Programming involves program development using a RAM processor and transference of the completed program to PROM. A **Numa-Logic** PROM burning unit is available as an option with the programmable controller if users are to burn their own PROM. Factory-supplied PROM burning services are also available.

6-3. COIL UTILIZATION

Output coils, either control relay (CR) or special function, can be programmed in a variety of ways. In any case, it is important to note that an effective program not only performs its function (although this is essential), but is also designed to increase efficiency and to decrease memory consumption. The programmer determines the manner in which techniques are applied to optimize both time and storage space. With practice, the programmer becomes adept at structuring programs for effectiveness. Some of the techniques used to achieve this objective are discussed in this section. They are:

- Dummy coils
- Oscillator circuits
- Transitional functions
- Programming order
- Multiple programs

6-4. DUMMY COILS

It might appear, at first, that the easiest method of developing a "dummy" coil (a coil that is never energized and whose contacts never change state) is simply not to program the coil. However, in practice, there are two conditions that make this impractical. The first is that the coil could be forced ON and with no logic to cause it to change state, remain ON when the force is deleted. The second drawback is that if someone else uses the program, an undocumented dummy coil could be programmed to serve another function, thereby, creating confusion and potential danger.

The best method for programming a dummy coil is illustrated in Figure 6-3. As shown, forcing coil CR0127 causes its contacts to change state. When the force is removed, the coil returns to the OFF state. In addition to consistently functioning properly, another benefit of using this circuit is that the coil is documented on any printout or drawing, preventing erroneous future use of the coil.

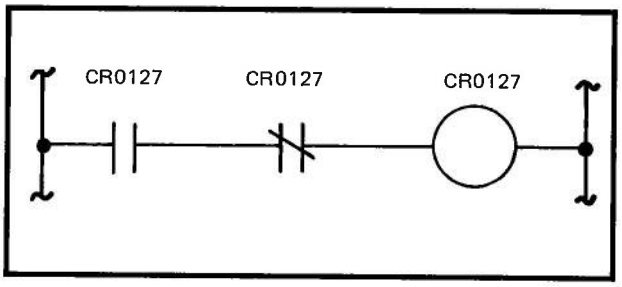

Figure 6-3. Dummy Coil Circuit

6-5. OSCILLATOR CIRCUITS

If an oscillator signal is required and the time interval is not critical, a simple oscillator can be programmed as illustrated in Figure 6-4. In this circuit, the scanning technique used in the processor causes the processor to see CR0126 as being OFF and the NC contacts of CR0126 as being closed (because CR0126 is OFF); it then energizes Coil CR0126. On the next scan, the processor sees Coil CR0126 as being ON, and CR0126 NC contacts as open (because Coil CR0126 was energized during the last scan); Coil CR0126 will then be turned OFF. The cycle then repeats itself. Coil CR0126 is ON every other scan.

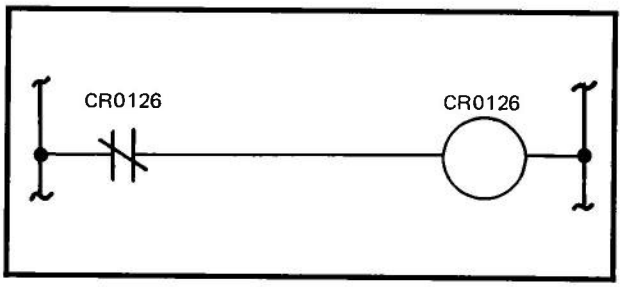

Figure 6-4. Oscillator Circuit

6-6. TRANSITIONAL FUNCTIONS

Many of the special and extended functions are transitional in operation. This means that a change in state of a set of contacts (non-conducting to conducting) is required to perform the function. Following are some examples of functions that are transitional in operation:

- Add
- Conversions
- Counters
- Divide
- Drum Controller
- Multiply
- Shift Registers
- Subtract

For example, in the case of counters, shift registers and drum controllers, transitional operation is a necessary and desired part of the function. However, in the case of the arithmetic and conversion function, this may or may not be desirable. If the processor does not have a Continuous Select (CS) function, scanning can be made to occur on a continuous (every scan) basis by using the format illustrated in Figure 6-5.

This form requires only that the second coil be that of a transitional function. In the example shown in Figure 6-5, UC0126 could be used. The assignment of operand and designation is unimportant, since the function is not used. (See paragraph 6-1 for a discussion on the HRRU and register manipulation.)

6-7. PROGRAM ORDER

Program order is generally not critical where a coil is placed in a particular program. However, when determining program order, instances do occur that require careful consideration of the location of a coil in a certain program. It is important to avoid the following situations, as improper program order can result in many hours of problem analysis.

Example 1

The circuit shown in Figure 6-6 illustrates one of these situations. Since Coil CR0059 is currently OFF, Coil CR0025 energizes via the CR0059 NC contacts. Coil CR0011 does not energize because the NO contact of CR0059 is open. Since Coil CR0025 is now energized, Coil CR0059 is energized. Scanning the circuit a second time finds the NC contacts of CR0059 now open, turning Coil CR0025 OFF. CR0011 does not energize because this time the NO contacts of CR0025 are open and Coil CR0059 turns OFF. As a result, CR0025 and CR0059 alternate ON/OFF, and CR0011 is never ON. The situation is demonstrated below:

Coil	Scan 1	Scan 2	Scan 3	Scan 4
CR0025	ON	OFF	ON	OFF
CR0011	OFF	OFF	OFF	OFF
CR0059	ON	OFF	ON	OFF

To change this situation to produce the desired result, all that is required is a change in the program order. Figure 6-7 illustrates what is required to correct the programming discussed above; the results of this program order are demonstrated below:

Coil	Scan 1	Scan 2	Scan 3	Scan 4
CR0025	ON	OFF	ON	OFF
CR0011	ON	OFF	ON	OFF
CR0059	ON	OFF	ON	OFF

Example 2

Another situation that requires attention to program order occurs in the use of output groups, as with a drum controller. In Figure 6-8, the drum controller is identified by the coil reference number 0001. Its destination is the OG0001. OG0001 consists of the discrete outputs 1 through 16. Placing the coil reference number in the same output group as the destination defeats the purpose. The drum controller will not run, as it is trying to control itself.

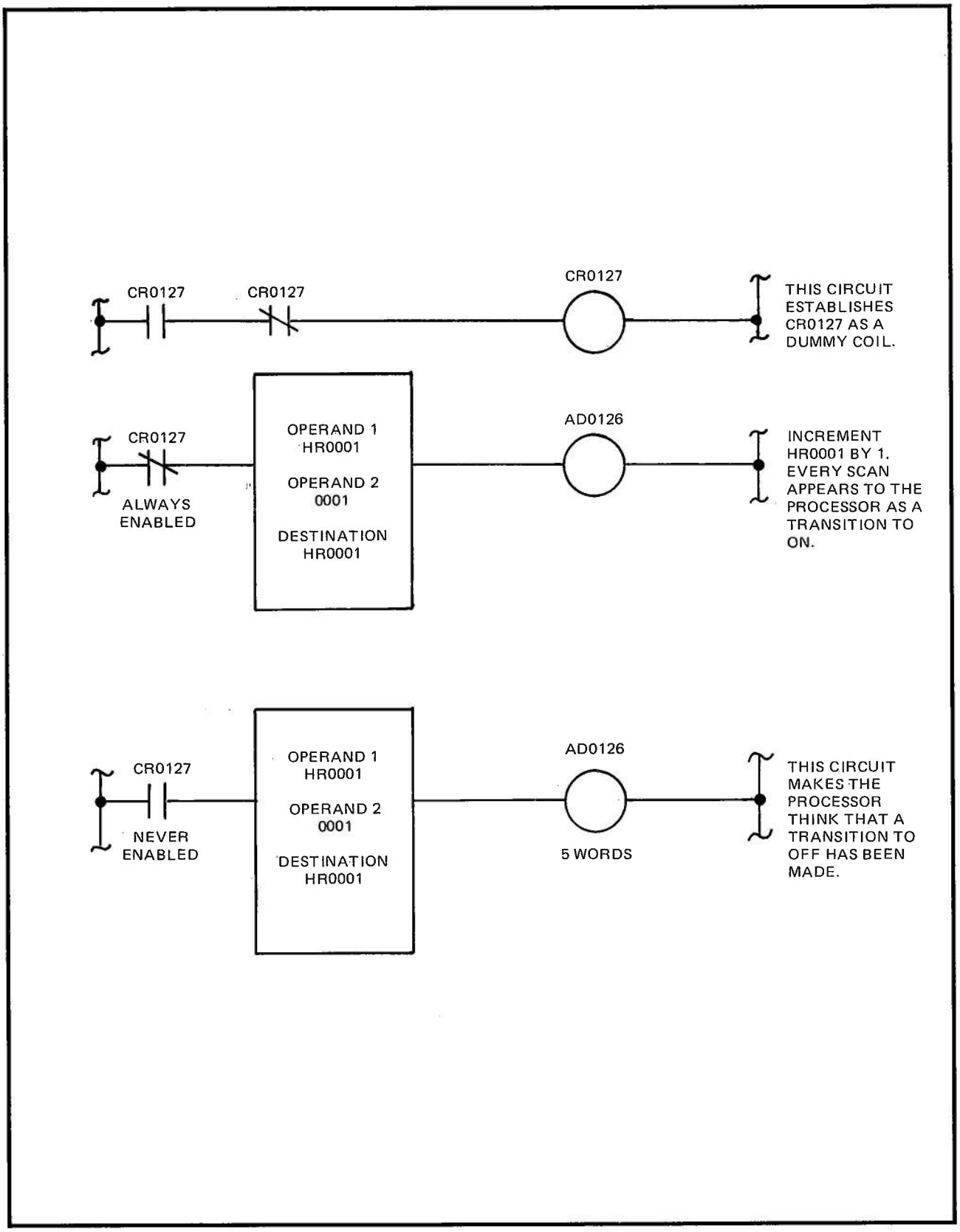

Figure 6-5. Every Scan Operation

6-8. MULTIPLE PROGRAMS

Using a single machine and a single control system, a **Numa-Logic** Programmable Controller based system is capable of performing a variety of sequences or operations on command. This capability is desirable in many instances. In the example shown in Figure 6-9, three separate Drum Controller (DR) functions are used to operate the same set of outputs (CR0017 through CR0032) through three separate sequences. It should be noted that this program could easily be expanded, with the drum controllers each assuming different lengths and each being stepped by an independent timing chain. This example has several features that are of interest:

- Every Master Control Relay (MR) is controlled by mutually-exclusive circuits (i.e., the logic is such that one and only one program can be selected at any given time).
- If no program is selected, all outputs (CR0017 through CR0032) are turned OFF.
- The functions are combined, when possible, into a single coil such as Rung 1 of the program.

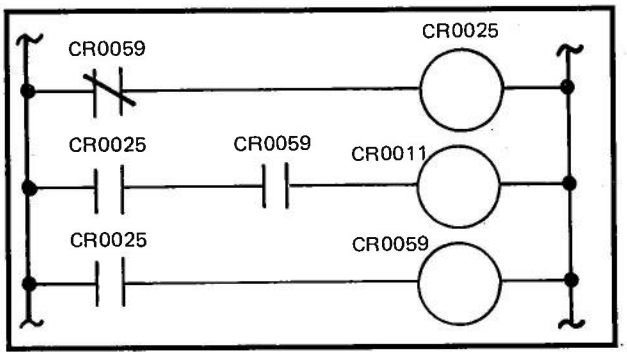

Figure 6-6. Improper Program Order

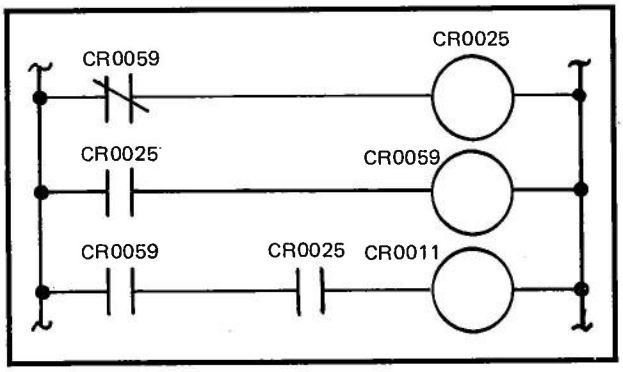

Figure 6-7. Correct Program Order

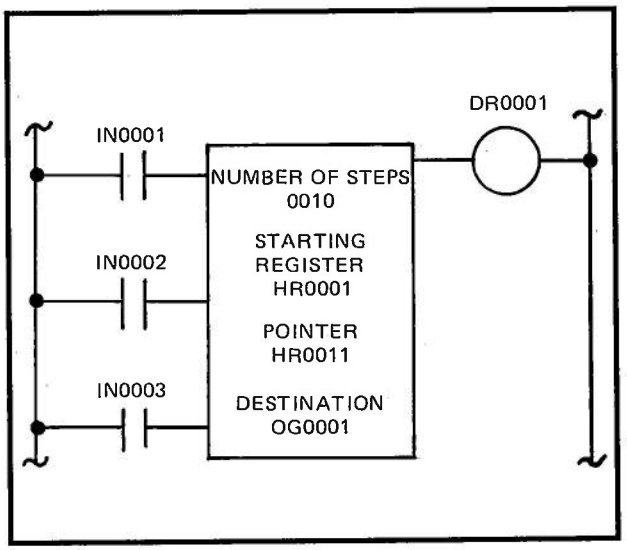

Figure 6-8. Drum Controller Pitfalls

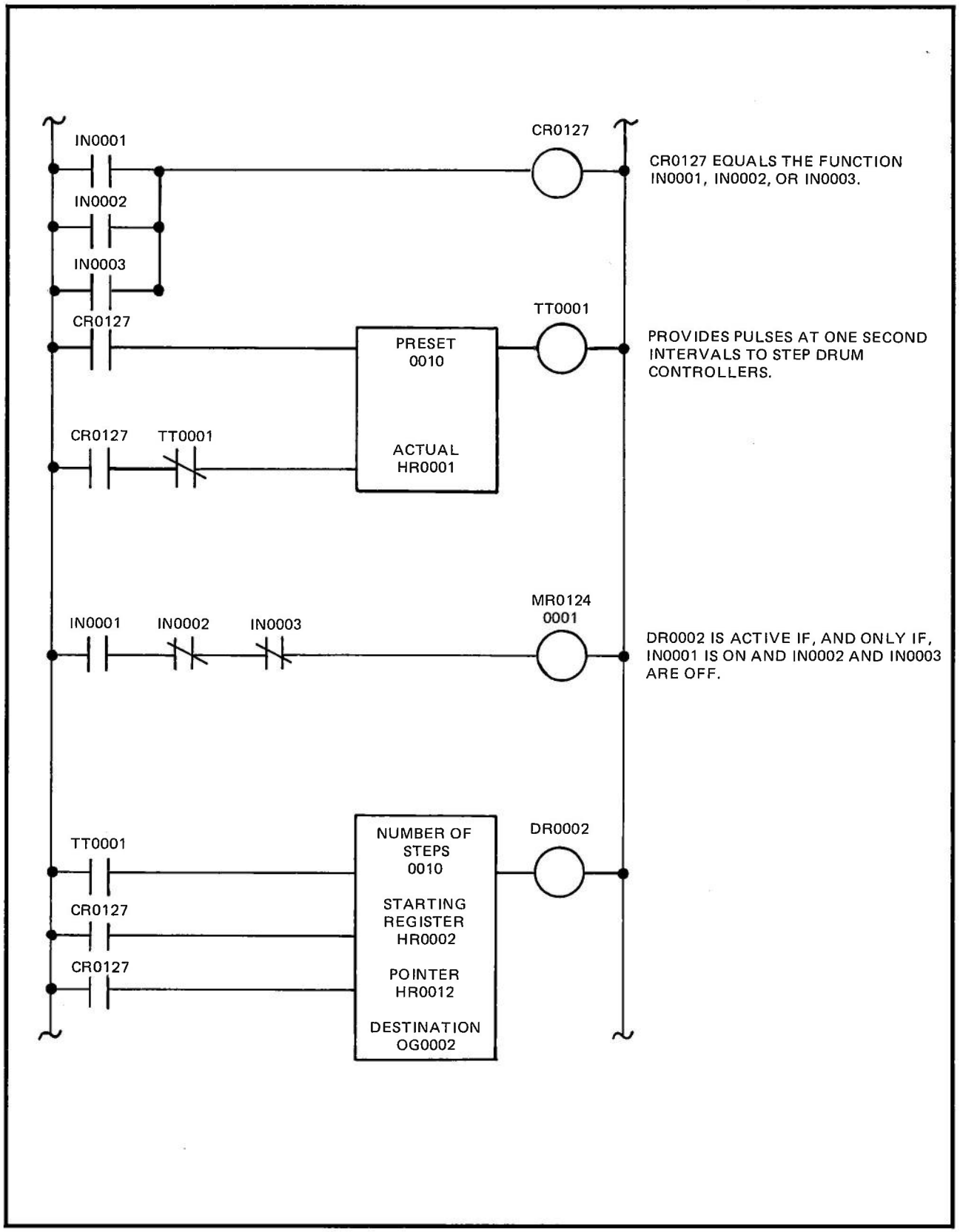

Figure 6-9a. Multiple Programs

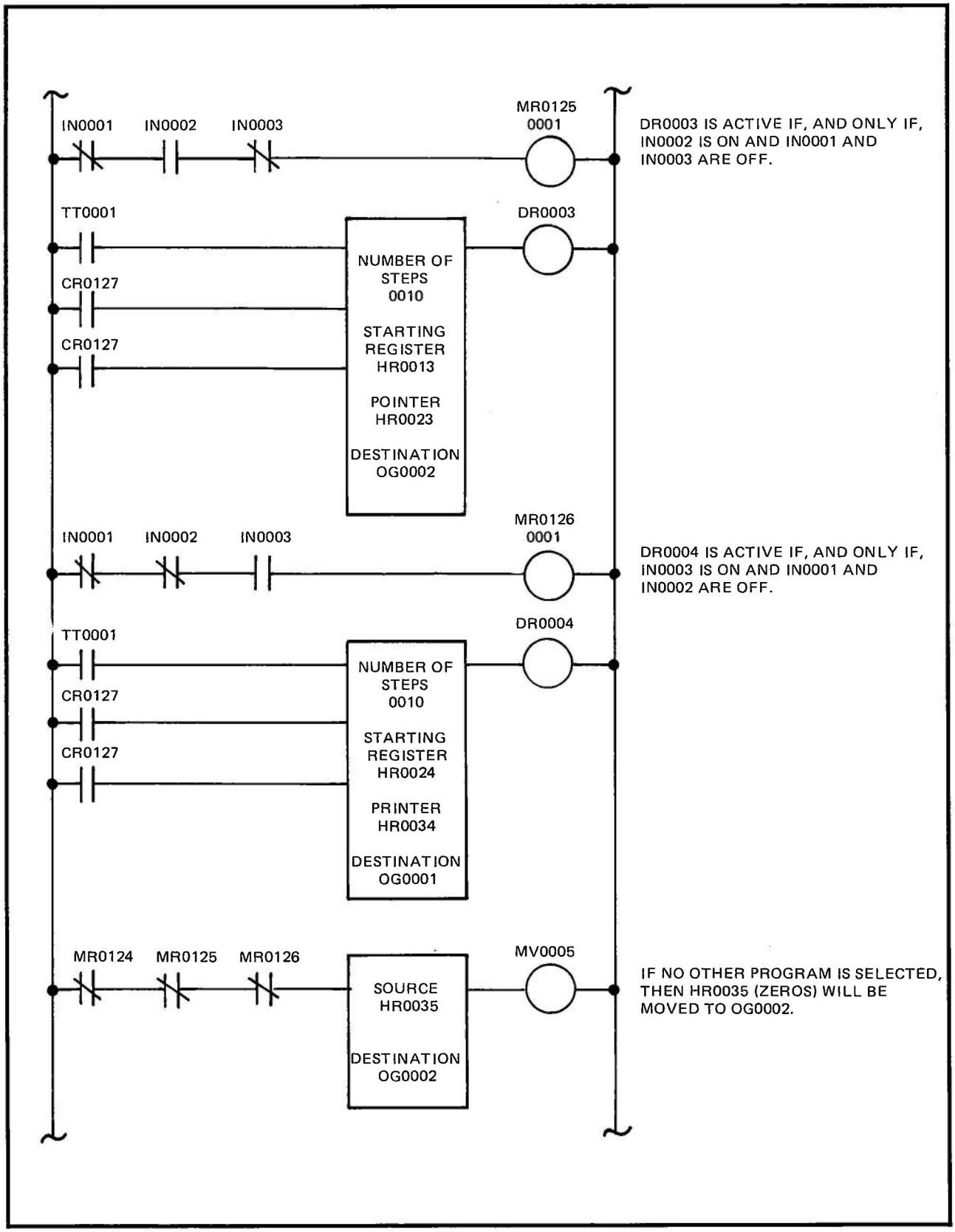

Figure 6-9b. Multiple Programs (Cont'd)