YPRT

Developing / patching the Printer ROM

Meindert Kuipers

Allschwill HP Summit, 2022

Contents

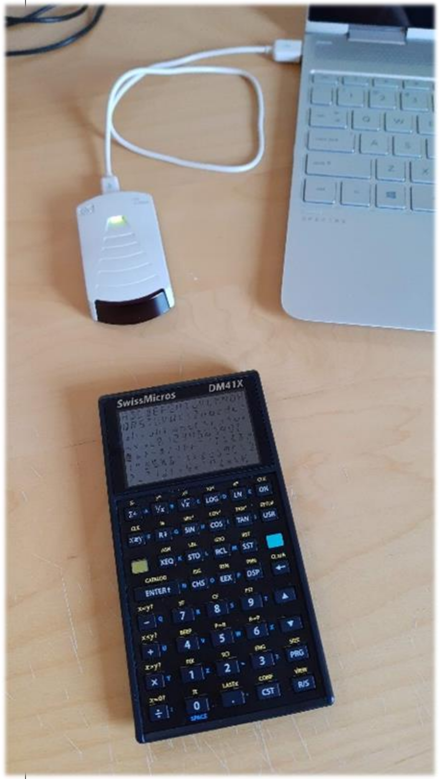
- Inspiration
- What is YPRT
- Studying the standard PRINTER 1E ROM
- Development method
- What must be done
- Plan A: just patching should be simple
- Plan B: It might be more complex then I thought
- Plan C: Help: Bankswitching needed
- What’s next

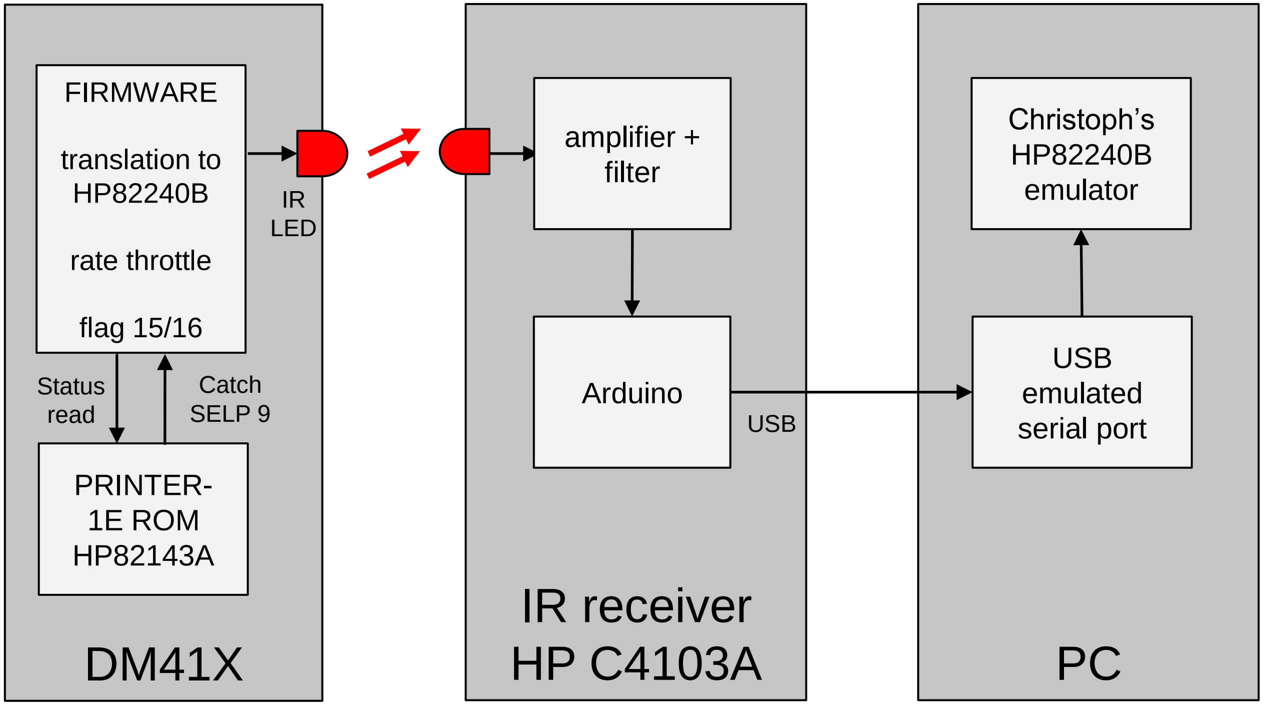
Inspiration

Inspiration came from the DM41X:

- The DM41X supports the standard and unpatched PRINTER-1E ROM to print to the infrared port to a real IR printer or IR receiver and printer simulator
- This is one of the few ways to get data out of the DM41X under HP41 program/mcode control, so it had my immediate interest
- I built a USB IR receiver (after the example of *Martin Hepperle*) which was my first ever Arduino project. This was great fun and works with the DM41X and WP34S (and probably with a real HP-41 IR module as well) and is expected to work with other IR based calculators.
- The PC-based HP82240 printer simulator (*Christoph*'s great piece of software) sees a serial port
- If this is possible with the DM41X, then it should be possible with the HP41CL

Inspiration





What is YPRT?

YPRT is an attempt to implement printing from a modified PRINTER-1E ROM to the serial port of the HP41CL

- To be used with a PC and printer emulator
- Or used by a real printer with serial port support
- Integrate in HP41CL ecosystem, Y**T was still available

Challenges

- Can flag 15/16 be used to control MAN / NORM / TRACE mode?
- Can we support graphics?
- Can it work with an HP82240B emulator?
- How to handle flags 21 and 55?

Studying the standard PRINTER 1E ROM

Must reads:

- HP82143A Printer Study by Doug Wilder (thanks Doug)
- HP82143A VASM Listings, improved
- PRINTER-1E disassembly listings
- HEPAX manual printer mcode section (and found a mistake)

Findings

- SELP 9 is the instruction to control the printer
- 6 possible commands
- There are blocks of NOP’s in the ROM, use these for extra code
- Printer status word contains printer buttons, mode and conditions
- The Saturn instruction set has a printer commands ?PBUSY that is never used

Studying the standard PRINTER 1E ROM

HP82143A printer commands:

- SELP 9 (0x264) transfers control to the HP82143A printer processor (Helios) with the following commands:

Mnemonic	Hex	Description
PFSET? 0 or BUSY?	0x003	Set carry if printer busy
PFSET? 2 or POWON?	0x083	Set carry if printer is ON
PFSET? 1 or VALID?	0x043	Set carry if status valid
PRINTC	0x007	Send byte in C[1:0] to the printer
RDPTRN	0x03A	Read the printer status word to C[13:10], must be followed by RTNCPU
RTNCPU	0x005	Return control to the HP41 CPU, must follow RDPTRN

Studying the standard PRINTER 1E ROM

Analysis of the SELP 9 printer commands:

- POWON?

  Used 9 times, various locations in the ROM

- VALID?

  Used 2 times, in the FNSTS routine (read printer status)

- RDPTRN, RTNCPU

  Used only once, in the FNSTS routine

- BUSY?

  Used twice, in the FNSTS and PBYTEC routine

- PRINTC

  Used only once in the PBYTEC routine

Studying the standard PRINTER 1E ROM

First conclusions:

- PBYTEC and FNSTS are the routines to be patched
- **POWON?** check will be simply removed
  - We simply assume that when YPRT is plugged in the HP41CL the user actually want to use this function, and we simply skip this test such that the ROM thinks the printer is always powered.
  - Therefore Flag 55 and Flag 21 will always be set after a power-on of the 41CL with YPRT plugged!
  - All occurances of POWON? are patched such that this always tests true
- **VALID?** test can be removed
  - Checks if the printer status is valid, since we intend to simulate the status we remove this test

Studying the standard PRINTER 1E ROM

First conclusions:

- **BUSY?** can be removed
  - Test is done when the status is checked, and a busy printer is not relevant in this case
  - More relevant is the BUSY check when sending data in PBYTEC, this must be replaced with a test if the serial port is ready and/or busy
- **RDPTRN** must be replaced
  - With a routine that simulates the status word by reading flags 15 and 16 and setting the correct other bits
- **PRINTC** must be replaced
  - With a routine that sends the byte to the HP41CL serial port
- **PRINTER-1E** function name has no RTN

Studying the standard PRINTER 1E ROM

Status bits:

Bit #	Mnemonic	Description	Use in Serial Printer ROM
15	SMA	TRACE mode when set	From User flag 15
14	SMB	NORM mode when set, MAN mode when 14 and 15 are clear	From User flag 16
13	PRT	PRINT key down	Not used, always 0
12	ADV	PAPER ADVANCE key down	Not used, always 0
11	OOP	Out Of Paper	Not used, always 0
10	LB	Low Battery	Not used, always 0
9	IDL	Idle condition	Not used, always 1
8	BE	Buffer Empty	Not used, always 1
7	LCA	Lower Case Alpha	Do we need to fill this? Comes from printer?
6	SCO	Special Column Output	Not used, always 0
5	DWM	Double Wide Mode	Not used, always 0
4	TEO	Type of End-Of-Line	Not used, always 0
3	EOL	Last End-Of-Line	Not used, always 0
2	HLD	Hold for Paper	Not used, always 0
1	-	Not used, always returns 1	Not used, always 0
0	-	Not used, always returns 1	Not used, always 0

Studying the standard PRINTER 1E ROM

Pitfalls:

- The PRINTER ROM is fixed in Page 6
- A fixed Page has the advantage that relative long GOSUB and GOTO are not needed
- Many entries are directly called from the HP41 OS or internal from the PRINTER ROM, these entries *must* be preserved
- The subroutines to be patched (FNSTS and PBYTEC) are cleverly programmed with a number of internal jumps and entries
- It is not always clear which registers are used
- It is not always clear how many subroutine levels may be used
- Several OS interrupt entries are used: PAUSE, I/O, ON, MEMORY LOST

Development method

What tools are best to create YPRT

- CalypsiNut is my favourite for developing a custom ROM image
  - Good assembler and linker
  - Results in a .rom file that can be loaded with a serial cable to the HP41CL
  - But the PRINTER ROM cannot be changed, entries must be preserved
  - I decided this would cause too much trouble to setup Calypsi in the proper way and would require a clean source of the PRINTER ROM for re-assembly
- Decision:
  - Use DAVID Assembler/HEXED to patch a copy of the PRINTER-1 E directly in a 41CL RAM page
  - Make regular backups of the ROM image to the PC

Development method

The challenge of documentation

- Replacing and patching a piece of original code requires careful tracking of addresses and registers
- No text editor entry with DAVID Assembler
- Turned to MS Excel for documenting code, and this turned to work out great with register tracking, especially when having to use bankswitching later

Development method

First experiences

- Editing a ROM in-situ in Page 6 is tricky
  - Especially when editing the entry points
  - Best edit in a neutral page with all entries disabled and plug in Page 6 when ready to test
  - Use an addition RAM page in Page 7 for the new routines
- Testing is most difficult:
  - Testing in a PC emulator is not an option due to the use of the 41CL serial port
  - No suitable debugging tools
  - Trial and error
  - Many crashes and need to remove the battery and do a full reset, and having to bootstrap the 41CL ☹ again and again, next project is a reset option for my HP41CL

What must be done

FNSTS routine – get the printer status

FNSTS - FETCH NEW STATUS

USES: C,ST[7:0],S9, NO PT, NO ADDITIONAL SUBROUTINE LEVELS

PRESERVES: ORIGINAL ST[7:0] IN C[1:0]

INPUT: S9=PRINTER INTERFACE ERROR FLAG

(IF S9=1 THEN NO ATTEMPT IS MADE TO READ STATUS)

OUTPUT: ORIGINAL ST[7:0] IS IN C[1:0]

IF S9=0, THEN FIRST BYTE OF PRINTER STATUS IS IN S[7:0] AND

SECOND BYTE OF PRINTER STATUS IS IN C[13:12]

ASSUMES: HEXMODE

FXSTS - FETCH EXISTING STATUS. SAME AS FNSTS EXCEPT DOESN’T SCRATCH

OLD STATUS BEFORE READING.

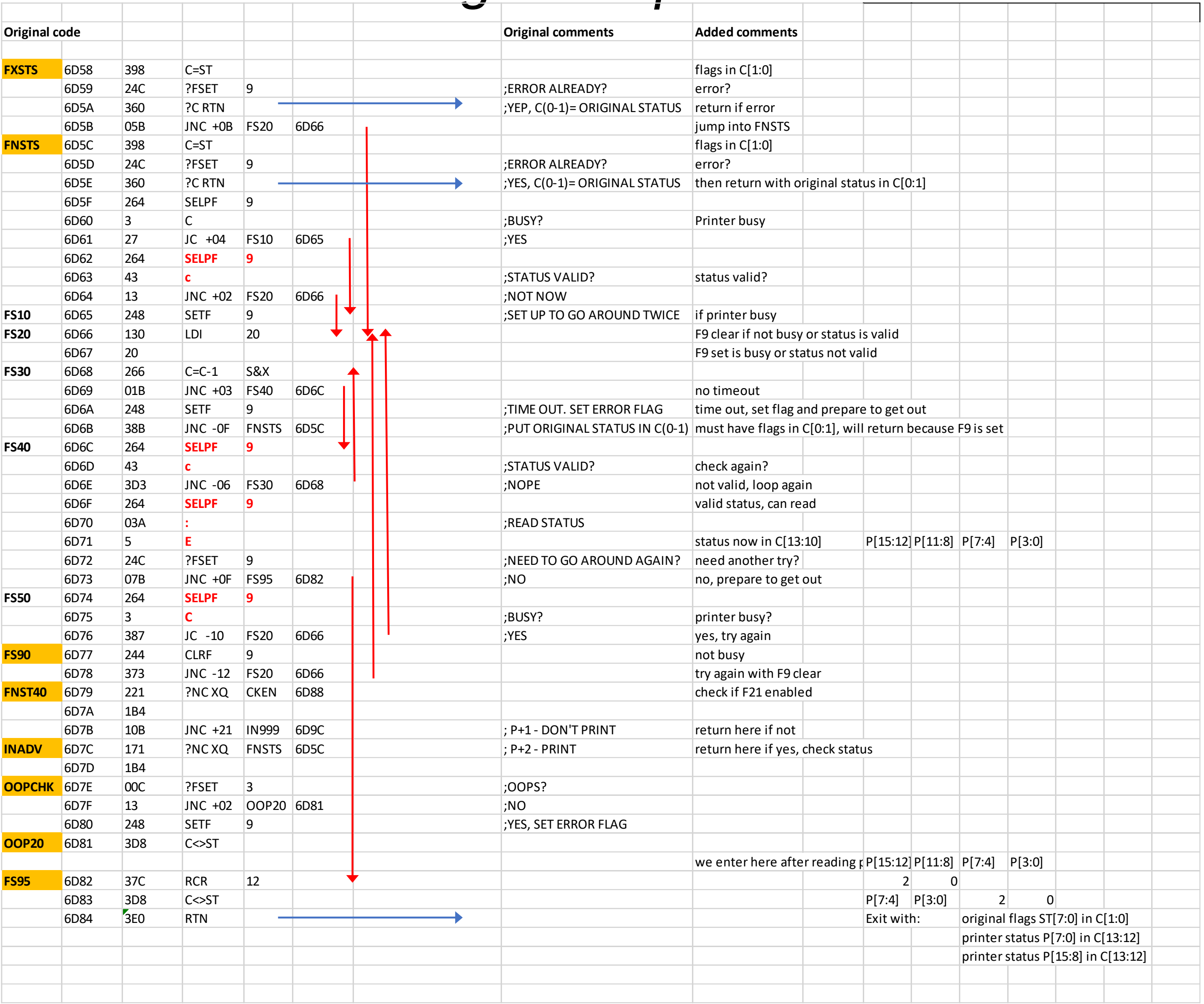
FS90 - FETCH EXISTING STATUS, ENTRY POINT FOR ERROR DIAGNOSIS ROUTINE.

SAME AS FXSTS EXCEPT IGNORES THE STATE OF S9 ON INPUT, DOESN’T

PRESERVE ORIGINAL ST[7:0], AND IGNORES PRINTER’S "BUSY" STATUS BIT

What must be done

FNSTS routine – get the printer status



Original code						Original comments	Added comments				
FXSTS	6D58	398	C=ST				flags in C[1:0]				
	6D59	24C	?FSET	9		;ERROR ALREADY?	error?				
	6D5A	360	?C RTN			;YEP, C(0-1)= ORIGINAL STATUS	return if error				
	6D5B	05B	JNC +0B	FS20	6D66		jump into FNSTS				
FNSTS	6D5C	398	C=ST				flags in C[1:0]				
	6D5D	24C	?FSET	9		;ERROR ALREADY?	error?				
	6D5E	360	?C RTN			;YES, C(0-1)= ORIGINAL STATUS	then return with original status in C[0:1]				
	6D5F	264	SELPF	9							
	6D60	3	C			;BUSY?	Printer busy				
	6D61	27	JC +04	FS10	6D65	;YES					
	6D62	264	SELPF	9							
	6D63	43	c			;STATUS VALID?	status valid?				
	6D64	13	JNC +02	FS20	6D66	;NOT NOW					
FS10	6D65	248	SETF	9		;SET UP TO GO AROUND TWICE	if printer busy				
FS20	6D66	130	LDI	20			F9 clear if not busy or status is valid				
	6D67	20					F9 set is busy or status not valid				
FS30	6D68	266	C=C-1	S&X							
	6D69	01B	JNC +03	FS40	6D6C		no timeout				
	6D6A	248	SETF	9		;TIME OUT. SET ERROR FLAG	time out, set flag and prepare to get out				
	6D6B	38B	JNC -0F	FNSTS	6D5C	;PUT ORIGINAL STATUS IN C(0-1)	must have flags in C[0:1], will return because F9 is set				
FS40	6D6C	264	SELPF	9							
	6D6D	43	c			;STATUS VALID?	check again?				
	6D6E	3D3	JNC -06	FS30	6D68	;NOPE	not valid, loop again				
	6D6F	264	SELPF	9			valid status, can read				
	6D70	03A	:			;READ STATUS					
	6D71	5	E				status now in C[13:10]	P[15:12]	P[11:8]	P[7:4]	P[3:0]
	6D72	24C	?FSET	9		;NEED TO GO AROUND AGAIN?	need another try?				
	6D73	07B	JNC +0F	FS95	6D82	;NO	no, prepare to get out				
FS50	6D74	264	SELPF	9							
	6D75	3	C			;BUSY?	printer busy?				
	6D76	387	JC -10	FS20	6D66	;YES	yes, try again				
FS90	6D77	244	CLRF	9			not busy				
	6D78	373	JNC -12	FS20	6D66		try again with F9 clear				
FNST40	6D79	221	?NC XQ	CKEN	6D88		check if F21 enabled				
	6D7A	1B4									
	6D7B	10B	JNC +21	IN999	6D9C	; P+1 - DON'T PRINT	return here if not				
INADV	6D7C	171	?NC XQ	FNSTS	6D5C	; P+2 - PRINT	return here if yes, check status				
	6D7D	1B4									
OOPCHK	6D7E	00C	?FSET	3		;OOPS?					
	6D7F	13	JNC +02	OOP20	6D81	;NO					
	6D80	248	SETF	9		;YES, SET ERROR FLAG					
OOP20	6D81	3D8	C<>ST								
							we enter here after reading	P[15:12]	P[11:8]	P[7:4]	P[3:0]
FS95	6D82	37C	RCR	12				2	0		
	6D83	3D8	C<>ST					P[7:4]	P[3:0]	2	0
	6D84	3E0	RTN					Exit with:		original flags ST[7:0] in C[1:0]	
										printer status P[7:0] in C[13:12]	
										printer status P[15:8] in C[13:12]	

What must be done

PBYTEC routine – send one byte to the printer

PBYTEC - SENDS A CONTROL BYTE TO THE PRINTER
ON ENTRY, C[1:0]=BYTE TO BE SENT TO THE PRINTER
AND S9=ERROR FLAG

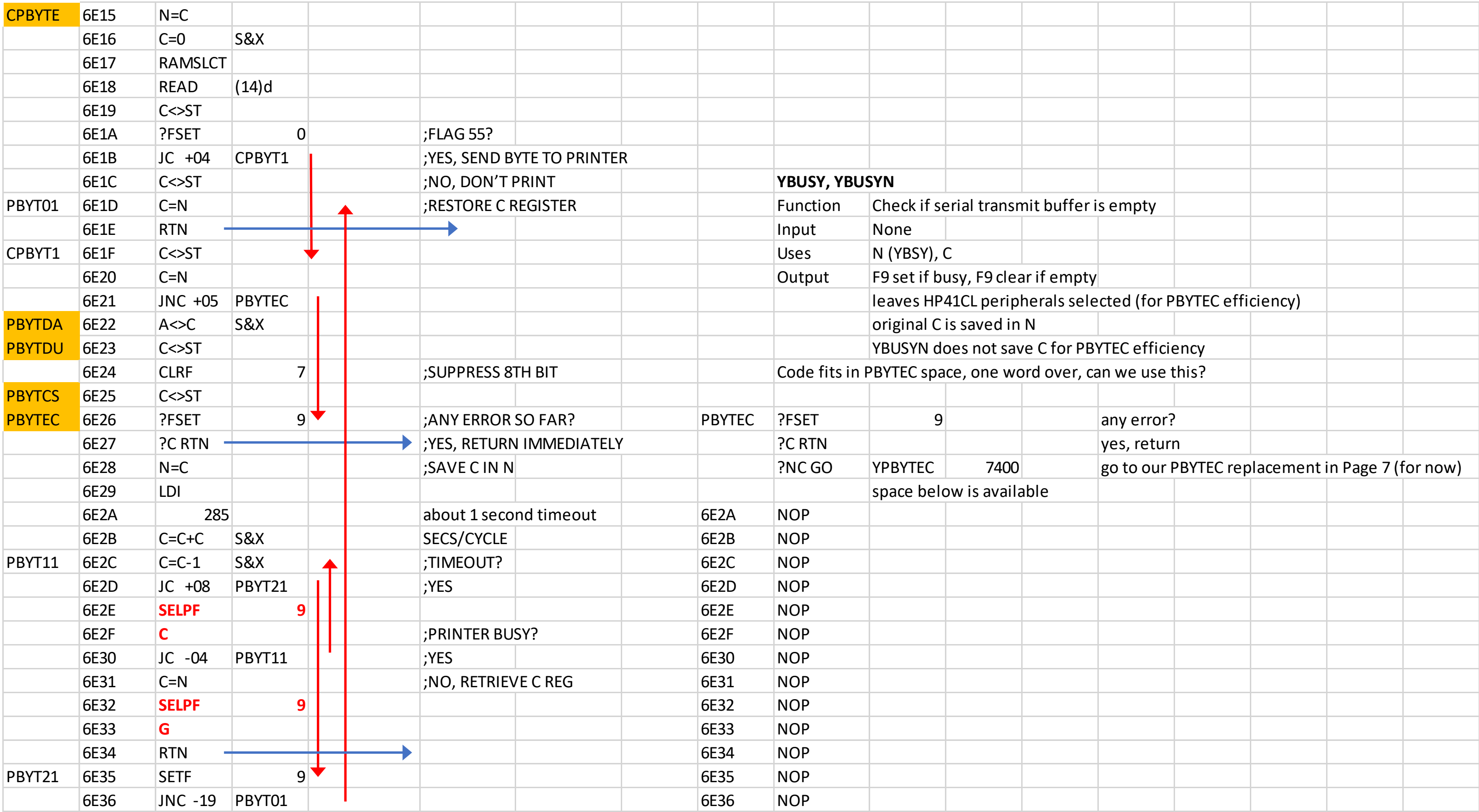
USES: N, NO PT, S9 FOR ERRORS, NO ADDITIONAL SUB LEVELS
IF S9=1 THEN DOES AN IMMEDIATE RETURN
WAITS UP TO 1 SECOND FOR THE PRINTER TO BE NOT BUSY. ON A TIMEOUT,
SETS S9 AND RETURNS.

PBYTDU - PRINT A BYTE OF DATA UNCONDITIONALLY. SAME AS PBYTEC
EXCEPT CLEARS BIT 7 OF THE DATA FRAME BEFORE SENDING IT TO THE
PRINTER.

CPBYTE - CONDITIONALLY PRINT BYTE. LOOKS AT FLAG 55 BEFORE DROPPING
INTO PBYTEC. IF FLAG 55 IS CLEAR, THEN DOES AN IMMEDIATE RETURN
WITHOUT SENDING ANYTHING TO THE PRINTER. USED FOR COUNTING
CHARACTERS TO SEE WHETHER THEY WILL FIT ON A LINE. FLAG 55 IS THE
PRINTER EXISTENCE FLAG, WHICH IS NOMINALLY ON ALL THE TIME THE
PRINTER IS PLUGGED IN.

What must be done

PBYTEC routine – send one byte to the printer

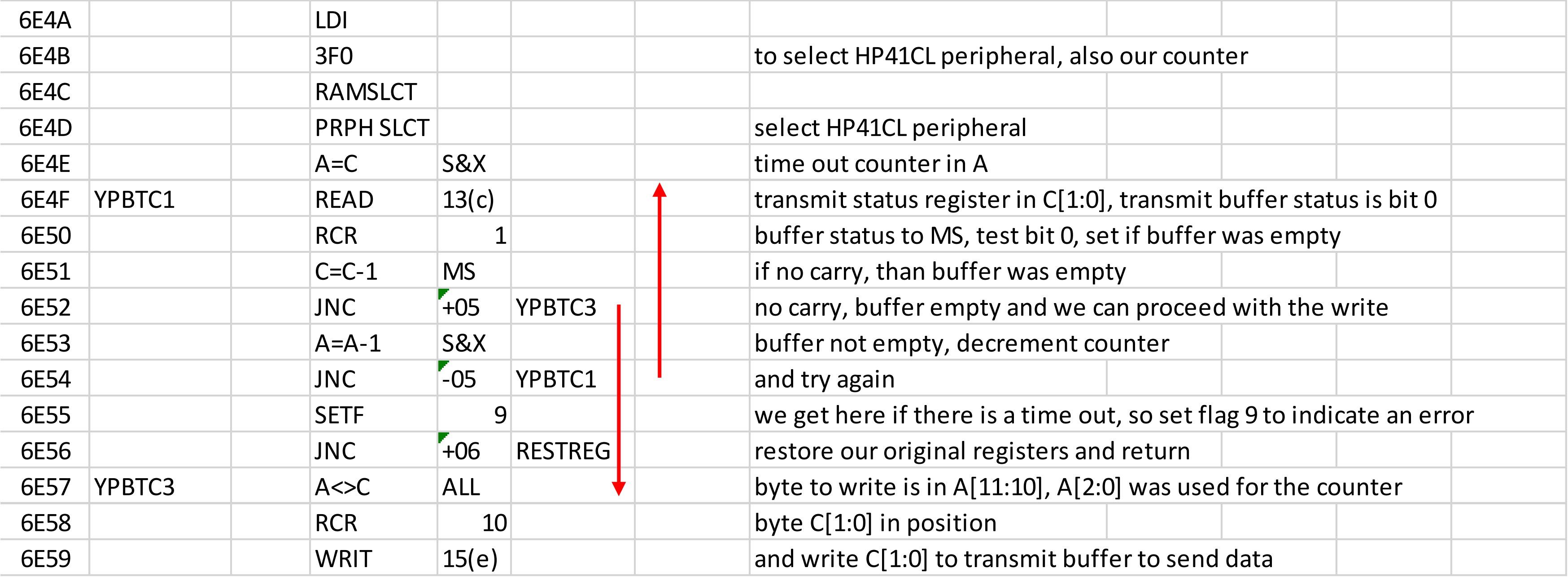


CPBYTE	6E15	N=C							
	6E16	C=0	S&X						
	6E17	RAMSLCT							
	6E18	READ	(14)d						
	6E19	C<>ST							
	6E1A	?FSET	0	;FLAG 55?					
	6E1B	JC +04	CPBYT1	;YES, SEND BYTE TO PRINTER					
	6E1C	C<>ST		;NO, DON'T PRINT		**YBUSY, YBUSYN**			
PBYT01	6E1D	C=N		;RESTORE C REGISTER		Function	Check if serial transmit buffer is empty		
	6E1E	RTN				Input	None		
CPBYT1	6E1F	C<>ST				Uses	N (YBSY), C		
	6E20	C=N				Output	F9 set if busy, F9 clear if empty		
	6E21	JNC +05	PBYTEC				leaves HP41CL peripherals selected (for PBYTEC efficiency)		
PBYTDA	6E22	A<>C	S&X				original C is saved in N		
PBYTDU	6E23	C<>ST					YBUSYN does not save C for PBYTEC efficiency		
	6E24	CLRF	7	;SUPPRESS 8TH BIT		Code fits in PBYTEC space, one word over, can we use this?			
PBYTCS	6E25	C<>ST							
PBYTEC	6E26	?FSET	9	;ANY ERROR SO FAR?	PBYTEC	?FSET	9		any error?
	6E27	?C RTN		;YES, RETURN IMMEDIATELY		?C RTN			yes, return
	6E28	N=C		;SAVE C IN N		?NC GO	YPBYTEC	7400	go to our PBYTEC replacement in Page 7 (for now)
	6E29	LDI					space below is available		
	6E2A	285		about 1 second timeout	6E2A	NOP			
	6E2B	C=C+C	S&X	SECS/CYCLE	6E2B	NOP			
PBYT11	6E2C	C=C-1	S&X	;TIMEOUT?	6E2C	NOP			
	6E2D	JC +08	PBYT21	;YES	6E2D	NOP			
	6E2E	SELPF	9		6E2E	NOP			
	6E2F	C		;PRINTER BUSY?	6E2F	NOP			
	6E30	JC -04	PBYT11	;YES	6E30	NOP			
	6E31	C=N		;NO, RETRIEVE C REG	6E31	NOP			
	6E32	SELPF	9		6E32	NOP			
	6E33	G			6E33	NOP			
	6E34	RTN			6E34	NOP			
PBYT21	6E35	SETF	9		6E35	NOP			
	6E36	JNC -19	PBYT01		6E36	NOP			

What must be done

Subroutine to send one byte to the 41CL serial port

- Not very complicated
- Select HP41CL peripheral
- Time-out loop to check if serial port is busy
- Transmit data



6E4A		LDI				
6E4B		3F0				to select HP41CL peripheral, also our counter
6E4C		RAMSLCT				
6E4D		PRPH SLCT				select HP41CL peripheral
6E4E		A=C	S&X			time out counter in A
6E4F	YPBTC1	READ	13(c)			transmit status register in C[1:0], transmit buffer status is bit 0
6E50		RCR	1			buffer status to MS, test bit 0, set if buffer was empty
6E51		C=C-1	MS			if no carry, than buffer was empty
6E52		JNC	+05	YPBTC3		no carry, buffer empty and we can proceed with the write
6E53		A=A-1	S&X			buffer not empty, decrement counter
6E54		JNC	-05	YPBTC1		and try again
6E55		SETF	9			we get here if there is a time out, so set flag 9 to indicate an error
6E56		JNC	+06	RESTREG		restore our original registers and return
6E57	YPBTC3	A<>C	ALL			byte to write is in A[11:10], A[2:0] was used for the counter
6E58		RCR	10			byte C[1:0] in position
6E59		WRIT	15(e)			and write C[1:0] to transmit buffer to send data

Plan A

Simplistic approach

- Patch all POWON? checks
- Replace all relevant SELP 9 instructions with a ?NCXQ to a new routine doing the actual operation to send the byte to the serial port
  - Only PRINTC remains to be implemented
- Use NOPs at 0x67DE (35 NOP’s)
- Patch the FNSTS routine to return a valid but fixed status word, ignore the flags for now
- Use NOPs at 0x67DE (35 NOP’s) for extra code
- Remove user code routine PRPLOT and PRPLOTP to make space for new routines, but first do some tests

Plan A

Results of Plan A

- Patching all POWON? checks was easy
- FNSTS routine would be much shorter
- All code might fit in this area and the NOPS area, may be able to keep user code in printer ROM

- First simple version actually worked, could PRA and ACA and PRP seemed to work
- But: PRX, PRSTK gave some strange results
- Printing of Display (AVIEW) gave strange results

- Further deep dive in the printer ROM:
  - All subroutine levels are used in PBYTEC

Plan B

Next step: do not use subroutines

- Not so complicated, do a few direct jumps and them jump back
- Issue was only in PBYTEC at first, but there was an issue in FNSTS as well when checking flags 15 and 16
- FNSTS issue was caused by conveniently doing a call to LDSST0 (get user flags in C), and resolved. Could would fit exactly in the space of FNSTS

- Fixing PBYTEC required to jump to the NOP space, and coded fitted just fine there

Plan B

Results of Plan B

- First version actually worked, could PRA and ACA and PRP seemed to work
- But: PRX, PRSTK still gave some strange results
- PRP seemed to work
- Printing of Display (AVIEW) gave strange results

- Further deep dive in the printer ROM:
  - Need to preserve nearly ALL registers in PBYTEC
  - Selecting the HP41CL peripheral deselects the display, and exactly that was needed for AVIEW and printing registers (PRX, PRSTK) to work

Plan C

Next step: need to preserve nearly all registers

- For correct implementation of serial write only one word needed to be saved, but where?
- Well, we have plenty of memory in the HP41CL, right?
- We do, but how to access and where?
- Solution found, but code to store something in HP41CL memory is tricky if almost everything else need to be saved, only N can be used
- Code is long, and would not fit: must use bankswitching
- Decided to use YUPS reserved space for intermediate storage

Plan C

Next step: fix the display selection

- A further dive in the printer ROM revealed the following:
  - Printing registers (not ALPHA) would print the display with the PRTLCD routine
- Fixing was easy, needed to include enabling the display in the main loop, and discovered that also here the C register needs to be preserved, and an alternative ENLCD routine was included

Final result

Finally everything worked

- Some additional patches were implemented
- To implement standard Bank Switching behavior some routines had to be patched
- There was some unexpected issues with polling entries. These were simply removed

- To patch a bankswitched ROM on the 41CL: simply plug the 2nd bank in its place *and* in a regular ROM page accessible to DAVID Assembler

- Now show some code

Next steps

We can only use Diego’s USB41 printer simulator

- Thanks to Diego that it exists
  - No graphics
- Would like to use Christophs HP82240B emulator
  - Graphics support

- Two possibilities
  - Ask Christoph to add the HP82143A printer to his software
  - Ask Diego to add graphics to his software
  - Create a YPRINT with a translation to the HP822240B printer
  - Patch the 82242A Infrared Printing Module

- All are technically possible

Next steps

Extending YPRT to support the HP82240B printer

- Fundamental differences exist
  - Could also patch the IR printer module?
    - No documentation about this module
    - No documentation about the SELPF 9 instructions
  - Extend existing YPRT to support 82240B?
    - The DM41X appears to do this translation
    - Need to store local state (graphics etc) in 41CL memory
    - For Column mode, need to store all colums or trick this (one column byte becomes 3 bytes to send)

- Now working with Christoph on the sources of his 82240 simulator to implement HP82143A support

QUESTIONS ?

Developing YPRT