HP-IL

Advanced HP-IL Communication

Advanced communication between HP-71B handheld computers over Hewlett-Packard Interface Loop.

by Sylvain Côté

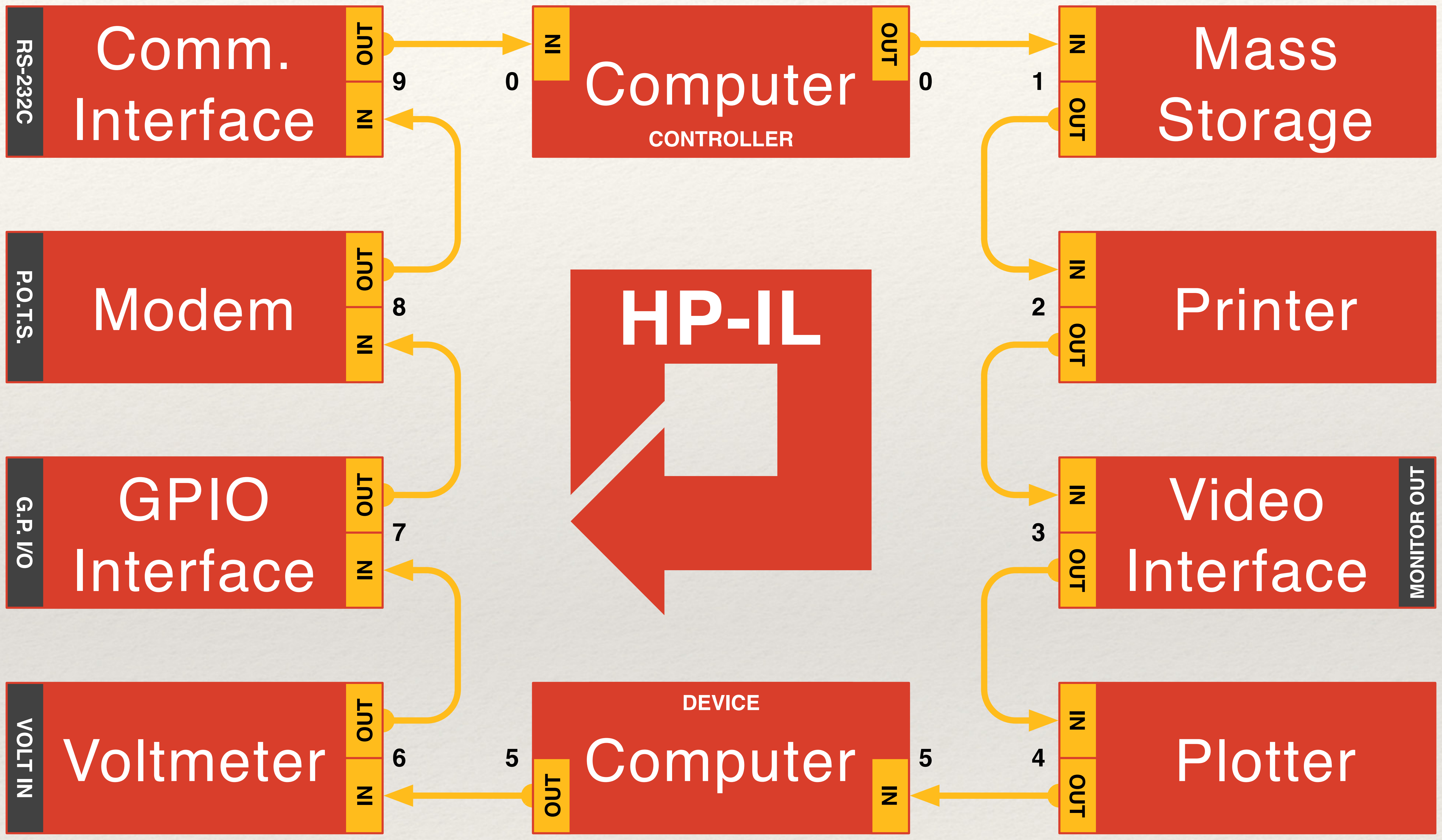
Table of Content

HP-IL

Introduction

Overview

- Low cost, low power, low speed, two wires serial interface.
- Loop (aka ring) network topology.
- Strongly inspired from Hewlett-Packard Interface Bus. (HP-IB, aka IEEE-488)
- Up to 31 devices can be connected to the loop using direct addressing or 961 devices when using extended addressing.



Actors

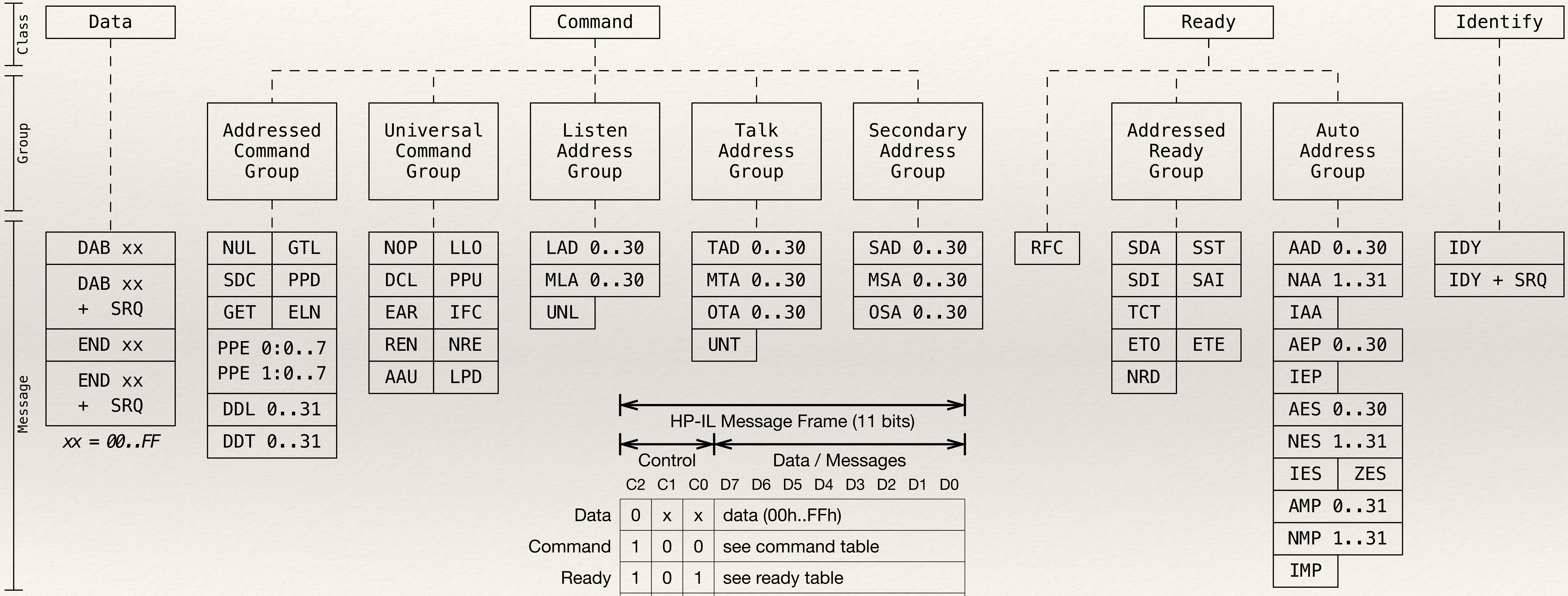
- Device
- A device ...
  - is a loop participant.
  - can be a listener and / or a talker.
  - can notify that it need to be serviced.

- Controller
- A controller ...
  - is the loop manager.
  - can be a listener and / or a talker.
  - can detect device service requests.
  - decide who talks and who listens.
- Only one controller per loop.

Roles

- Listener
  - Listener role is assigned by the controller.
  - Multiple listeners can exist at the same time in a loop.
  - Listeners receive data.
  - Listeners receive DDL requests. (Device Dependent Listener)
- Talker
  - Talker role is assigned by the controller.
  - Only one talker can exist at any given time in a loop.
  - Talker send data.
  - Talker receive DDT requests. (Device Dependent Talker)

Messages



	C2	C1	C0	D7 D6 D5 D4 D3 D2 D1 D0
Data	0	x	x	data (00h..FFh)
Command	1	0	0	see command table
Ready	1	0	1	see ready table
Identify	1	1	x	0 or parallel poll value

Messages Tables

Command & Ready

Command (4xx) Messages

	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	
0 0000	NUL	GTL			SDC	PPD			GET							ELN	0 0000
1 0001	NOP	LLO			DCL	PPU			EAR								1 0001
2 0010	LAD 0	LAD 1	LAD 2	LAD 3	LAD 4	LAD 5	LAD 6	LAD 7	LAD 8	LAD 9	LAD 10	LAD 11	LAD 12	LAD 13	LAD 14	LAD 15	2 0010
3 0011	LAD 16	LAD 17	LAD 18	LAD 19	LAD 20	LAD 21	LAD 22	LAD 23	LAD 24	LAD 25	LAD 26	LAD 27	LAD 28	LAD 29	LAD 30	UNL	3 0011
4 0100	TAD 0	TAD 1	TAD 2	TAD 3	TAD 4	TAD 5	TAD 6	TAD 7	TAD 8	TAD 9	TAD 10	TAD 11	TAD 12	TAD 13	TAD 14	TAD 15	4 0100
5 0101	TAD 16	TAD 17	TAD 18	TAD 19	TAD 20	TAD 21	TAD 22	TAD 23	TAD 24	TAD 25	TAD 26	TAD 27	TAD 28	TAD 29	TAD 30	UNT	5 0101
6 0110	SAD 0	SAD 1	SAD 2	SAD 3	SAD 4	SAD 5	SAD 6	SAD 7	SAD 8	SAD 9	SAD 10	SAD 11	SAD 12	SAD 13	SAD 14	SAD 15	6 0110
7 0111	SAD 16	SAD 17	SAD 18	SAD 19	SAD 20	SAD 21	SAD 22	SAD 23	SAD 24	SAD 25	SAD 26	SAD 27	SAD 28	SAD 29	SAD 30		7 0111
8 1000	PPE 0.0	PPE 0.1	PPE 0.2	PPE 0.3	PPE 0.4	PPE 0.5	PPE 0.6	PPE 0.7	PPE 1.0	PPE 1.1	PPE 1.2	PPE 1.3	PPE 1.4	PPE 1.5	PPE 1.6	PPE 1.7	8 1000
9 1001	IFC		REN	NRE							AAU	LPD					9 1001
A 1010	DDL 0	DDL 1	DDL 2	DDL 3	DDL 4	DDL 5	DDL 6	DDL 7	DDL 8	DDL 9	DDL 10	DDL 11	DDL 12	DDL 13	DDL 14	DDL 15	A 1010
B 1011	DDL 16	DDL 17	DDL 18	DDL 19	DDL 20	DDL 21	DDL 22	DDL 23	DDL 24	DDL 25	DDL 26	DDL 27	DDL 28	DDL 29	DDL 30	DDL 31	B 1011
C 1100	DDT 0	DDT 1	DDT 2	DDT 3	DDT 4	DDT 5	DDT 6	DDT 7	DDT 8	DDT 9	DDT 10	DDT 11	DDT 12	DDT 13	DDT 14	DDT 15	C 1100
D 1101	DDT 16	DDT 17	DDT 18	DDT 19	DDT 20	DDT 21	DDT 22	DDT 23	DDT 24	DDT 25	DDT 26	DDT 27	DDT 28	DDT 29	DDT 30	DDT 31	D 1101
E 1110																	E 1110
F 1111																	F 1111
	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	

Ex.: 400=NUL, 49A=AAU, 490=IFC, 422=LAD 02, 43F=UNL, 441=TAD 01, 45F=UNT, …

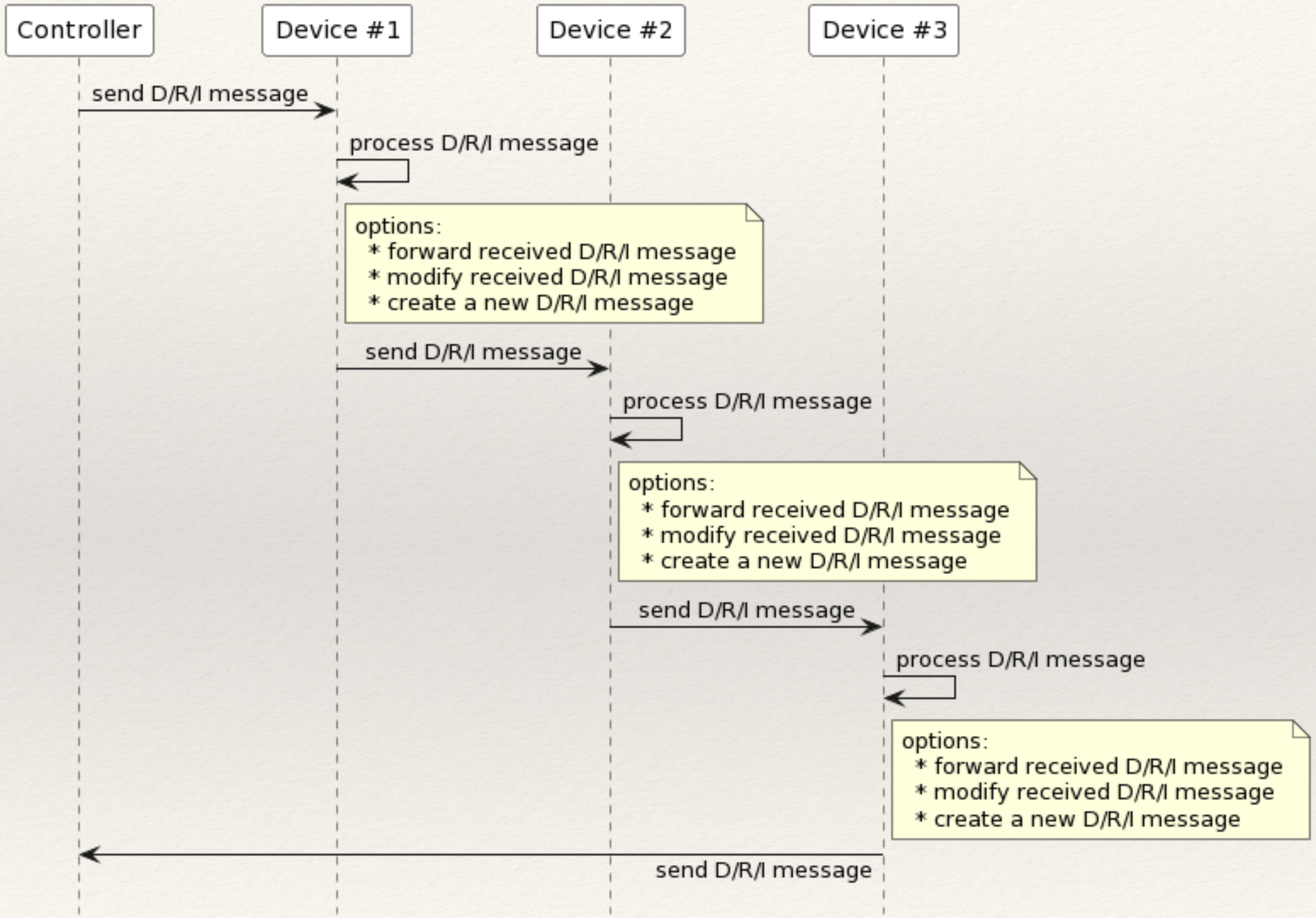
Ready (5xx) Messages

	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	
0 0000	RFC																0 0000
1 0001																	1 0001
2 0010																	2 0010
3 0011																	3 0011
4 0100	ETO	ETE	NRD														4 0100
5 0101																	5 0101
6 0110	SDA	SST	SDI	SAI	TCT												6 0110
7 0111																	7 0111
8 1000	AAD 0	AAD 1	AAD 2	AAD 3	AAD 4	AAD 5	AAD 6	AAD 7	AAD 8	AAD 9	AAD 10	AAD 11	AAD 12	AAD 13	AAD 14	AAD 15	8 1000
9 1001	AAD 16	AAD 17	AAD 18	AAD 19	AAD 20	AAD 21	AAD 22	AAD 23	AAD 24	AAD 25	AAD 26	AAD 27	AAD 28	AAD 29	AAD 30	IAA	9 1001
A 1010	AEP 0	AEP 1	AEP 2	AEP 3	AEP 4	AEP 5	AEP 6	AEP 7	AEP 8	AEP 9	AEP 10	AEP 11	AEP 12	AEP 13	AEP 14	AEP 15	A 1010
B 1011	AEP 16	AEP 17	AEP 18	AEP 19	AEP 20	AEP 21	AEP 22	AEP 23	AEP 24	AEP 25	AEP 26	AEP 27	AEP 28	AEP 29	AEP 30	IEP	B 1011
C 1100	AES 0	AES 1	AES 2	AES 3	AES 4	AES 5	AES 6	AES 7	AES 8	AES 9	AES 10	AES 11	AES 12	AES 13	AES 14	AES 15	C 1100
D 1101	AES 16	AES 17	AES 18	AES 19	AES 20	AES 21	AES 22	AES 23	AES 24	AES 25	AES 26	AES 27	AES 28	AES 29	AES 30	IES	D 1101
E 1110	AMP 0	AMP 1	AMP 2	AMP 3	AMP 4	AMP 5	AMP 6	AMP 7	AMP 8	AMP 9	AMP 10	AMP 11	AMP 12	AMP 13	AMP 14	AMP 15	E 1110
F 1111	AMP 16	AMP 17	AMP 18	AMP 19	AMP 20	AMP 21	AMP 22	AMP 23	AMP 24	AMP 25	AMP 26	AMP 27	AMP 28	AMP 29	AMP 30	IMP	F 1111
	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	

Ex.: 500=RFC, 540=ETO, 542=NRD, 560=SDA, 561=SST, 563=SAI, 583=AAD 3, …

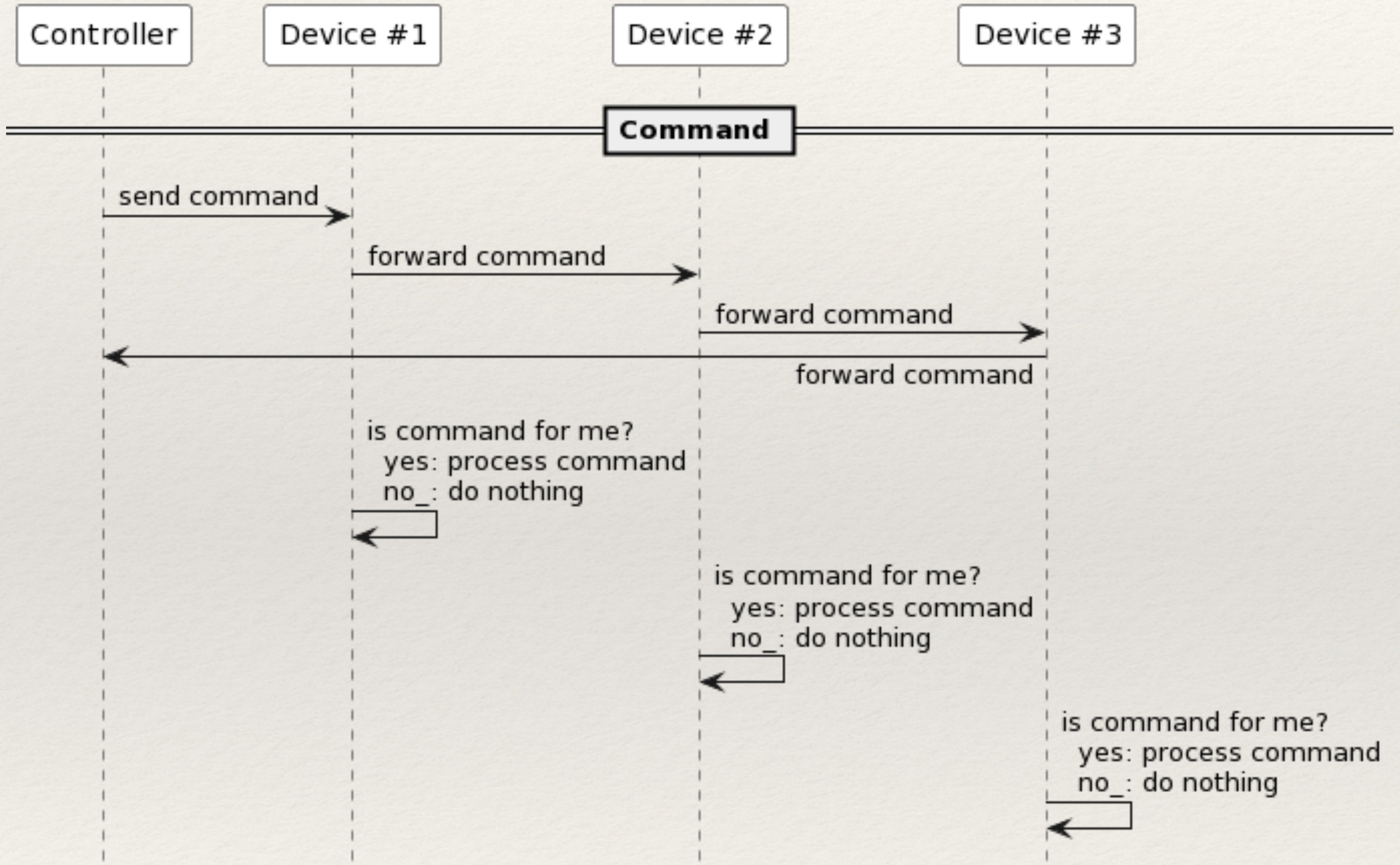
Communication Pattern

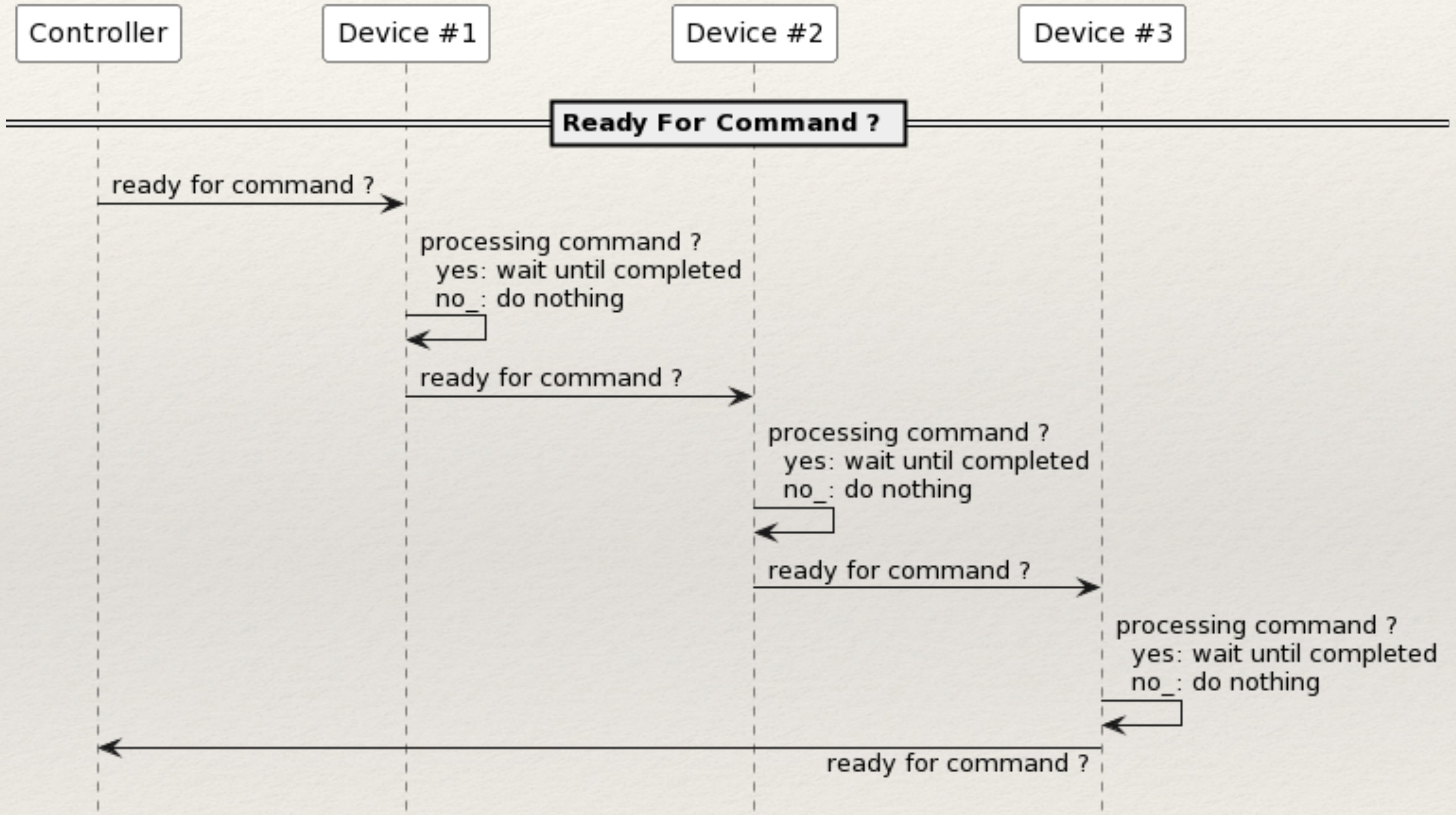
Data / Ready / Identify



Communication Pattern

Command





Equipment

Hardware & Software

- Physical Devices
  - 71B 2+ BASIC Handheld Computer.
  - 82401A 2+ HP-71B HP-IL Interface.
  - 82167x 2+ HP-IL cable. (x→ A:0.5m, B:1m, D:5m)
  - 82402A opt. HP-71B Dual HP-IL Adapter.
- Virtual Devices
  - Computer To run emulators & simulators.
  - PIL-Box HP-IL / USB Interface from Jean-François Garnier.
  - Virtual IL HP-IL devices simulators from Christoph Gießelink. (Win32)
  - pyILPER HP-IL devices simulators from Joachim Siebold. (Python)

HP 71B Handheld Computer

- The most advanced interface loop controller ever made that can also act as a device if so desired.
- Can be configured with three IL interfaces using two dual IL adapter.
  - 82401A HP-IL Interface Module
  - 82402A Dual HP-IL Adapter
    (can host two 82401A interface modules)
- Identifiers: ID = "HP71" & AID = 3

Why using a 71 instead of a 41 or a 75 ?

- Although all of these machines (41, 71 & 75) can be used as a controller, only the 71 has the capability to be configured as a device.
- This capability allows several 71s to coexist on the same loop, thus opening the possibility to have: coordinated distributed computing, peer-to-peer messaging, remote control, etc.
- Another interesting feature, because the 71 IL module has his own communication processor, the IL module can do basic communication while the 71 is sleeping and can wake up the 71 main processor if more advanced communication handling is needed.
- Last but not least, the possibility to have up to three HP-IL modules in the same 71 allow it to simultaneously take the role of controller or device in one loop and the role of controller or device in another loop.

Language Extension Files

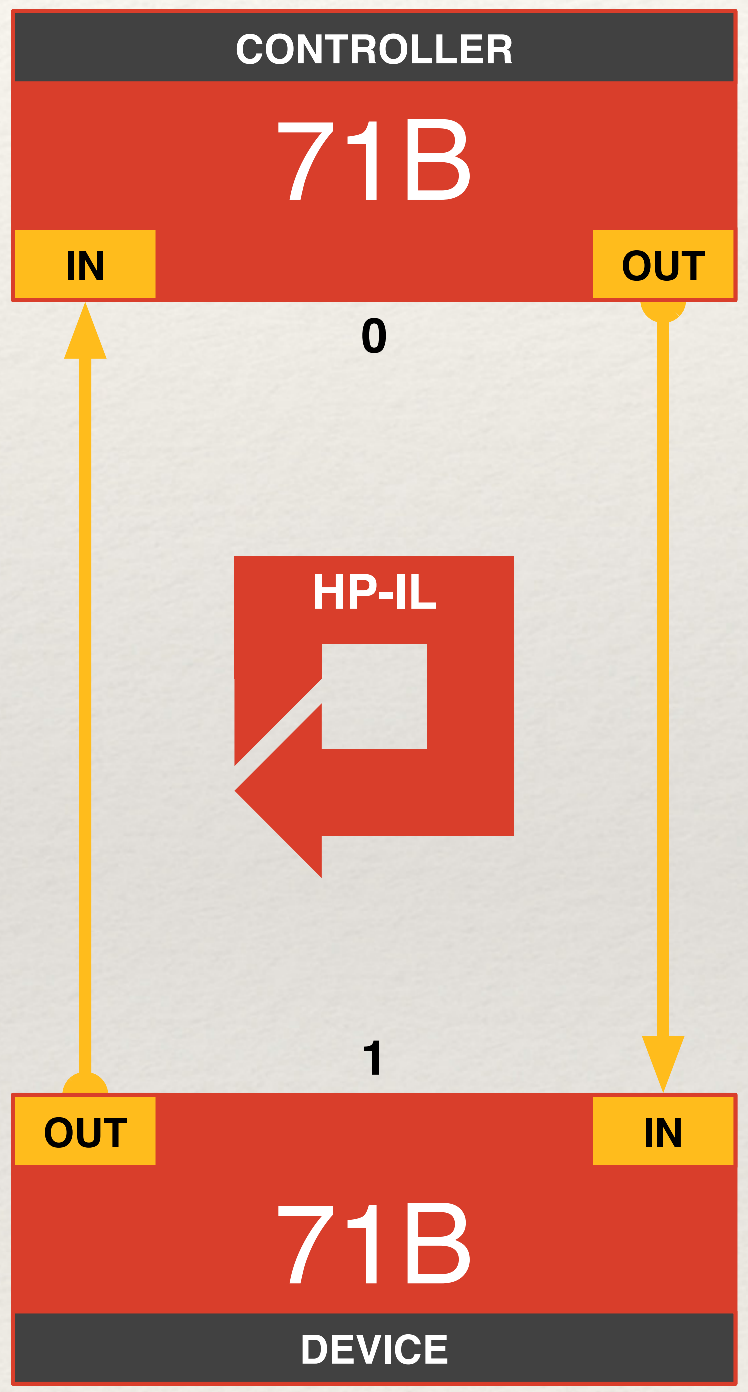
- NLOOPLEX — LEX file from Jean-François Garnier.
  - NLOOP() — Return the number of addressable devices in the loop.
  - PPOLL() — Return the data portion of a IDY/ISR IL frame.
  - SLEEP — Put the CPU to sleep.
  - SRQ() — Return the C0 ctrl bit (SRQ) value of a IDY/ISR IL frame.
- XHPIL — LEX file, author unknown.
  - ADDRESS — Return the number of addressable devices in the loop.
  - COUNTIN — ?
  - MYADDR — Return my device loop address.
  - PPOLL() — Return the data portion of a IDY/ISR IL frame.

Communication I

One Controller
One Device

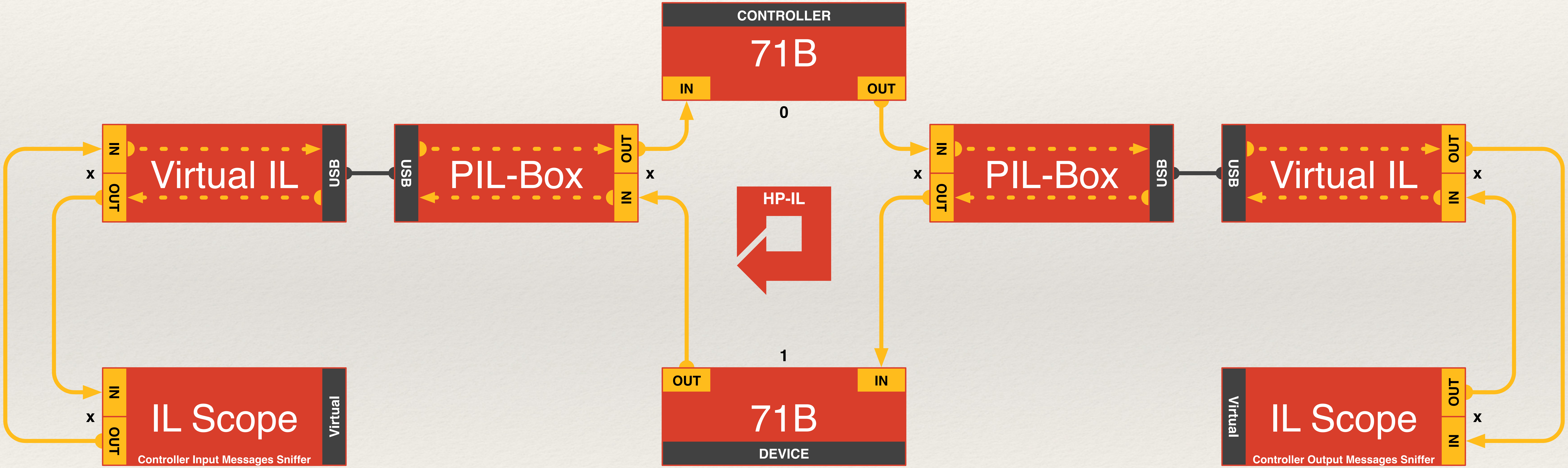
Loop Configuration 1

One controller with one device



Loop Configuration 1

One controller with one device and two frames sniffers



Device Status

Device Status Byte(s)

...	Byte 0							
	7	6	5	4	3	2	1	0
	128	64	32	16	8	4	2	1

7	1: System	0: Device
6	1: SRQ	0: Normal
5 – 0	Value	

```
10000001b:81h Sys. Low Bat.
11000001b:C1h Sys. Low Bat.    + SRQ
00000001b:01h Device specific
01000001b:41h Device specific + SRQ
```

System Status Byte Values

Stat	+SRQ	Events
81h	C1h	Low Battery
82h	C2h	Manual Intervention Required
83h	C3h	Data Error
84h	C4h	Device Error
85h	C5h	No room for data
86h	C6h	Self Test Failure
87h	C7h	Command Error
88h	C8h	Powering Down
89h	C9h	External Service Request
8Ah	CAh	Device Dependent Service Request
9Fh	DFh	ASCII Follows

Stat	+SRQ	States
80h	C0h	All okay
A0h	E0h	Request loop control
A1h	E1h	Ready to receive data
A2h	E2h	Ready to send data
A3h	E3h	Not ready to receive/send data

Setting & Getting Device Status

Controller

```
CONTROL ON                    ! 71B controller → address=0
                              ! 71B device     → address=1
```

Abbreviations: Controller Output Message, Controller Input Message.

```
IL Scope: CONTROL ON
Ctrl.Out.Msg | Ctrl.Inp.Msg | Comment
IFC    (490) | IFC    (490) | interface clear
RFC    (500) | RFC    (500) | ready for command ?
AAU    (49A) | AAU    (49A) | auto address unconfigure
RFC    (500) | RFC    (500) | ready for command ?
AAD 1  (581) | AAD 2  (582) | auto address assignment
```

Device

```
REQUEST 31                    ! set device status (31d = 1Fh)
```

Controller

```
S = SPOLL(1)                  ! get status from device @ address 1
DISP S                        ! LCD: "31"
```

Abbreviations: Controller Output Message, Controller Input Message.

```
IL Scope: SPOLL(1)
Ctrl.Out.Msg | Ctrl.Inp.Msg | Comment
UNL    (43F) | UNL    (43F) | unlisten
RFC    (500) | RFC    (500) | ready for command ?
TAD 01 (441) | TAD 01 (441) | talk address #1
RFC    (500) | RFC    (500) | ready for command ?
SST    (561) | DAB 1F (01F) | send status → status
DAB 1F (01F) | ETO    (540) | status → end of transmission ok
UNT    (45F) | UNT    (45F) | untalk
RFC    (500) | RFC    (500) | ready for command ?
UNL    (43F) | UNL    (43F) | unlisten
RFC    (500) | RFC    (500) | ready for command ?
```

Device

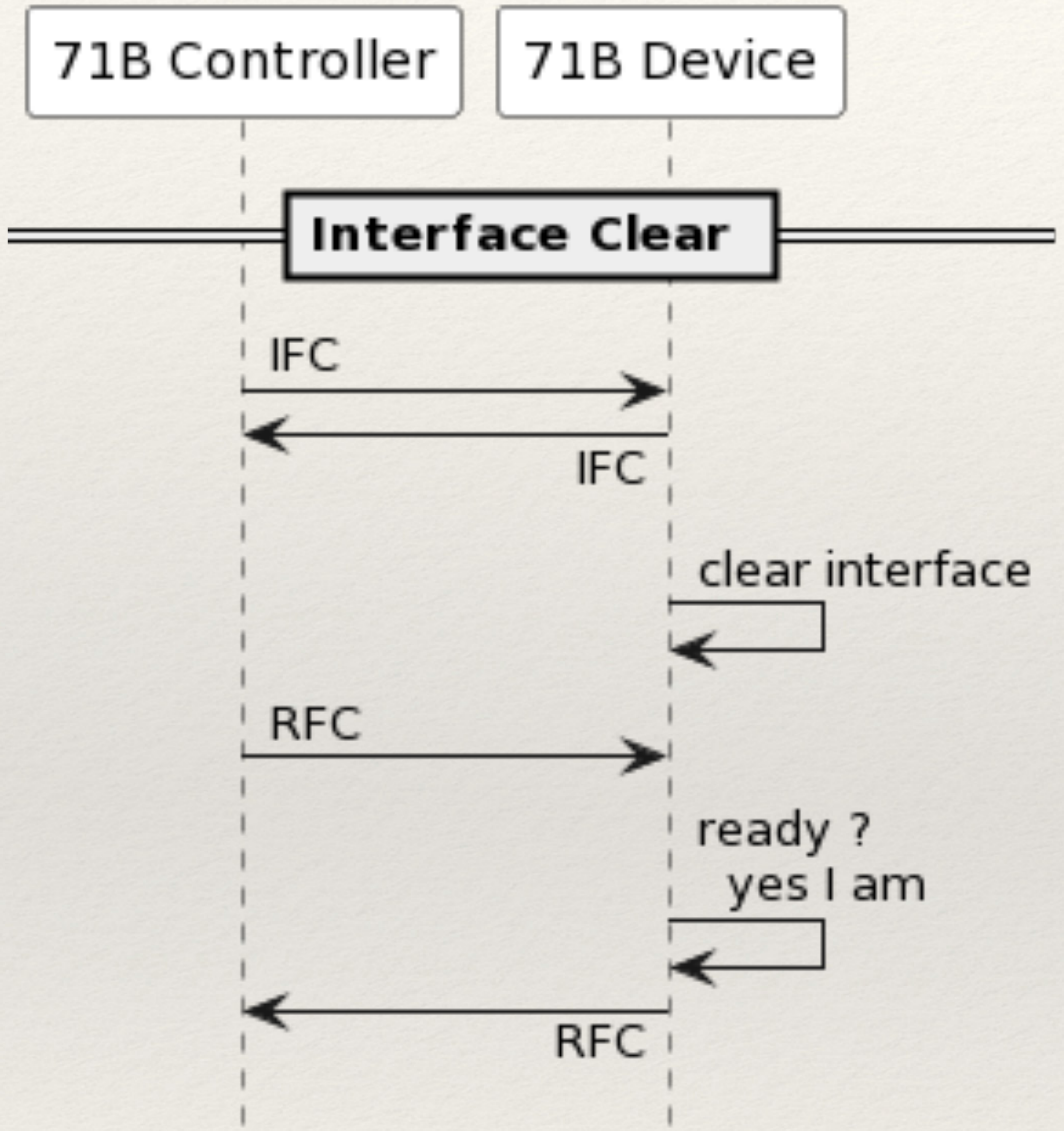
```
REQUEST 0                     ! reset device status (00d = 00h)
```

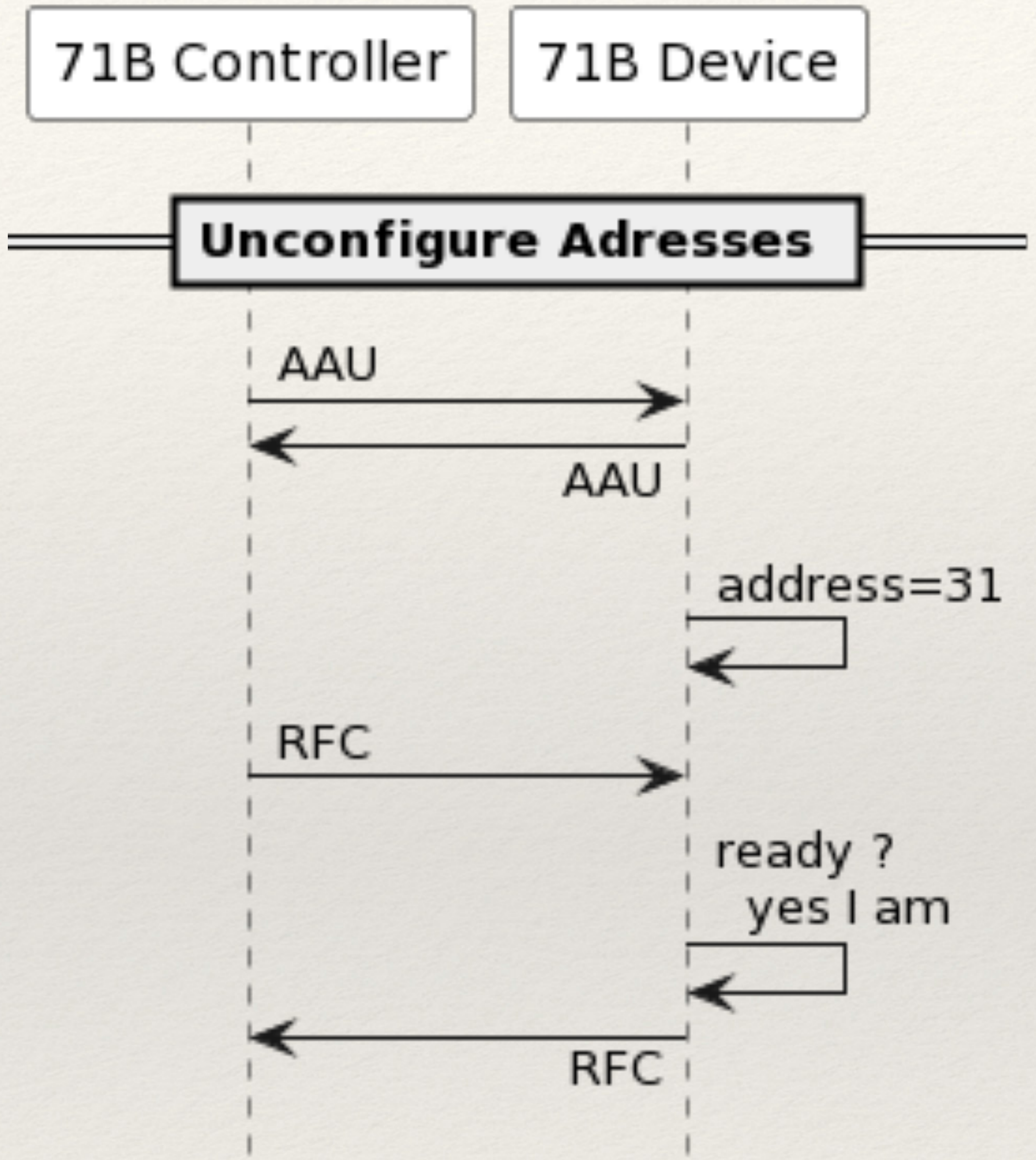
Controller

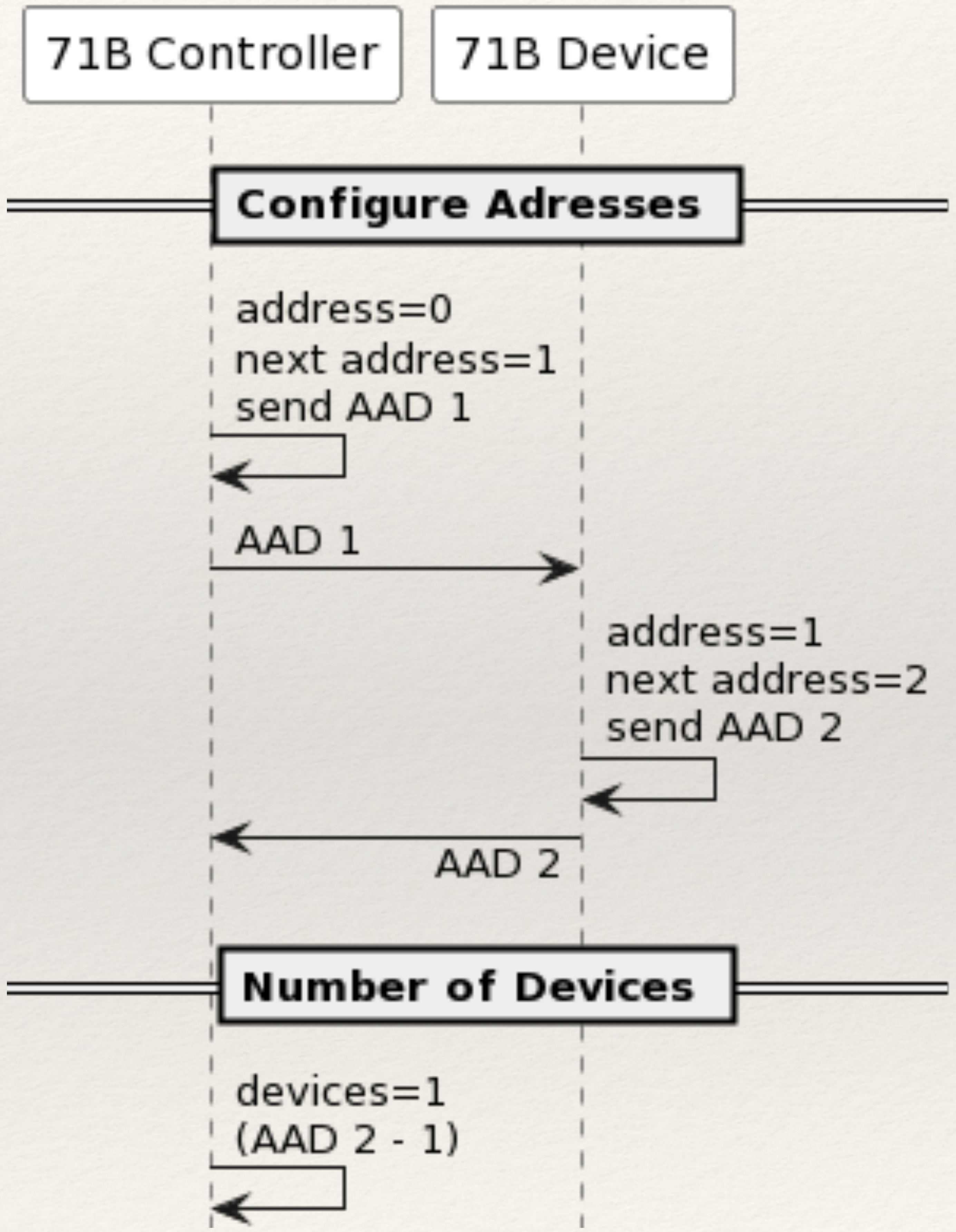
```
S = SPOLL(1)                  ! get status from device @ address 1
DISP S                        ! LCD: "0"
```

CONTROL ON

Sequence Diagram

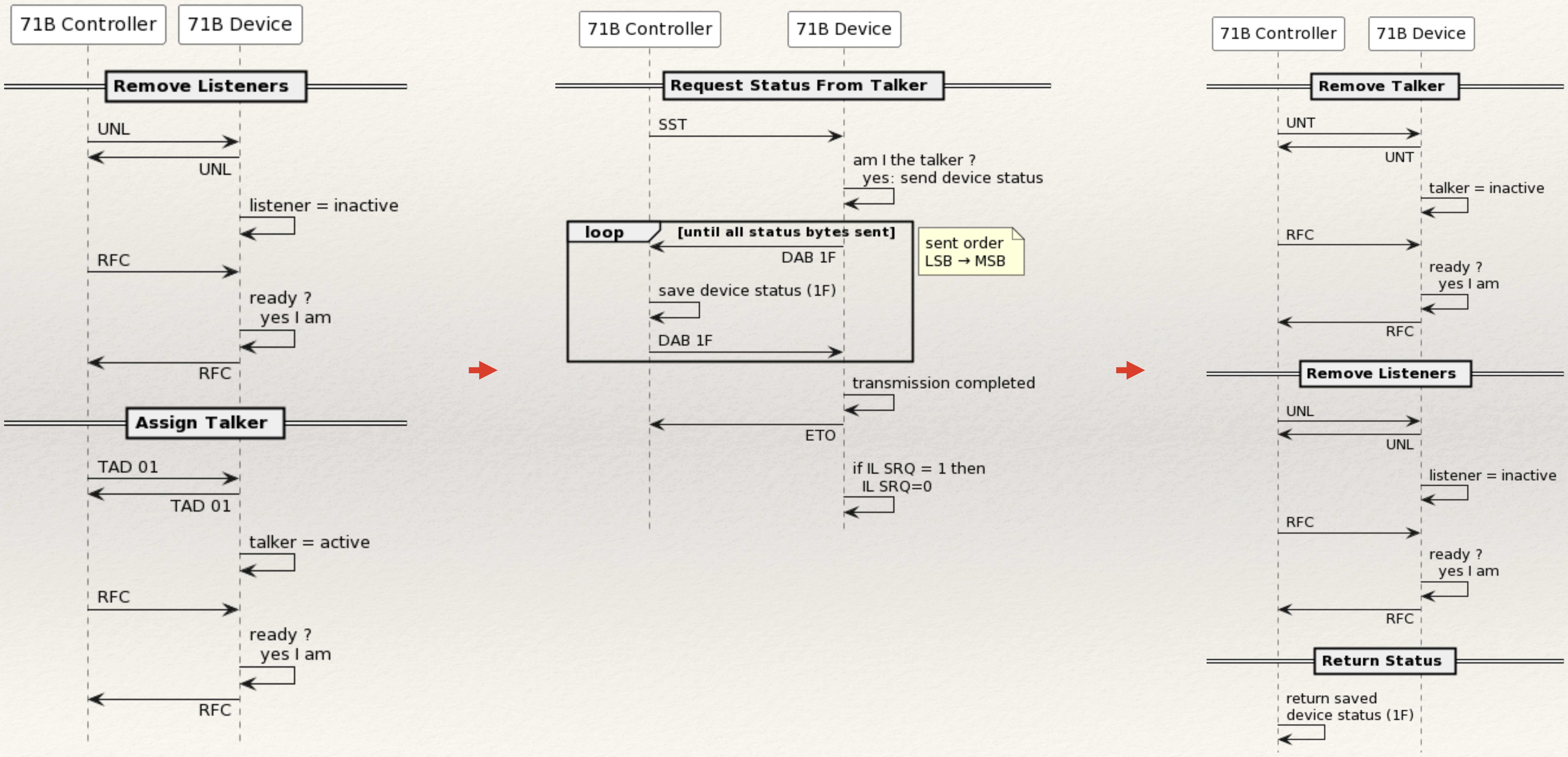




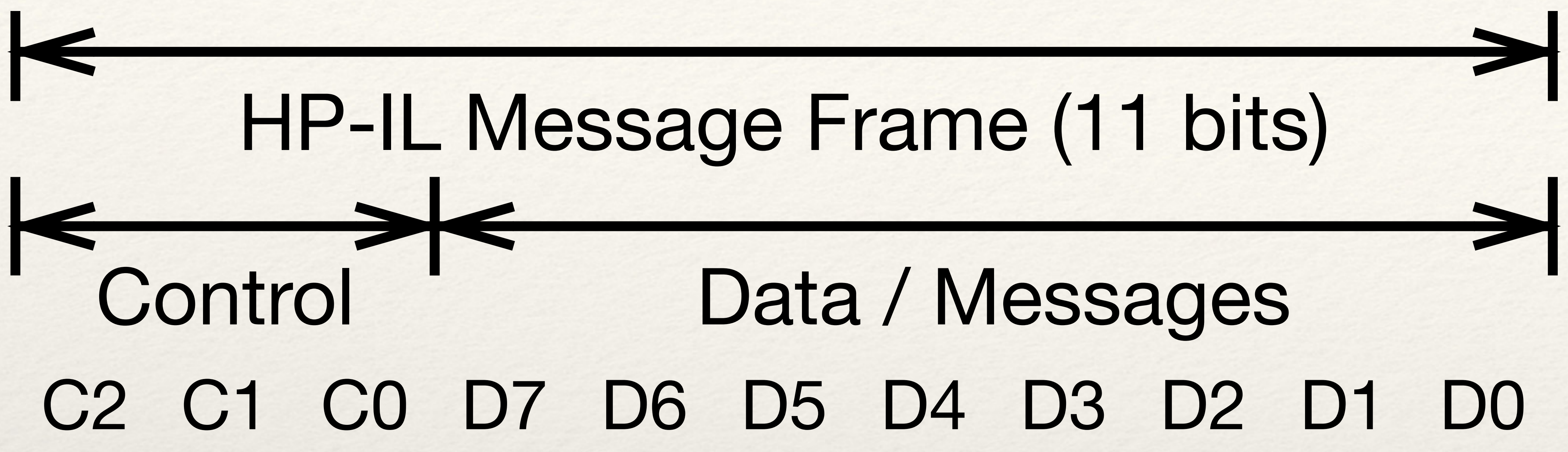


S = SPOLL(1)

Sequence Diagram



Service Request



C2	C1	C0	D7–D0
0	0	x	Data
0	1	x	Data [end of record]
1	0	0	Command
1	0	1	Ready
1	1	x	Identify

SRQ service request bit

Device Status Byte(s)

…	Byte 0							
	7	6	5	4	3	2	1	0
	128	64	32	16	8	4	2	1
	sys	srq	status					

- Device Status[LSB][bit 6] alias: DS-SRQ
- On new Device Status → IL-SRQ = DS-SRQ
- On SST → IL-SRQ = 0
- On Data / IDY → Control[C0] = IL-SRQ

Service Request Detection

Single Device

```
Controller
CONTROL ON                   ! 71B controller → address=0
                             ! 71B device     → address=1
```

```
Device
REQUEST 1                    ! set IL-SRQ=0, DS-SRQ=0, DS-Sts=1
                            ! 01d = 01h = 00000001b → 0:0:000001
```

```
Controller
SEND IDY                     ! set C0=IL-SRQ (0)
```

```
Abbreviations: Controller Output Message, Controller Input Message.
IL Scope: SEND IDY
Ctrl.Out.Msg | Ctrl.Inp.Msg | Comment
IDY     (600) | IDY     (600) | identify (C0=0)
```

```
Device
REQUEST 64                   ! set IL-SRQ=1, DS-SRQ=1, DS-Sts=0
                            ! 64d = 40h = 01000000b → 0:1:000000
```

```
Controller
SEND IDY                     ! set C0=IL-SRQ (1)
SEND IDY                     ! set C0=IL-SRQ (1)
```

```
Abbreviations: Controller Output Message, Controller Input Message.
IL Scope: SEND IDY
Ctrl.Out.Msg | Ctrl.Inp.Msg | Comment
IDY     (600) | ISR     (700) | interrupt service request (C0=1)
IDY     (600) | ISR     (700) | interrupt service request (C0=1)
```

Service Request Handling

Single Device

```
CONTROL ON                         ! 71B controller → address=0
                                   ! 71B device     → address=1
```
Controller

↓

```
REQUEST 66                         ! set IL-SRQ=1, DS-SRQ=1, DS-Sts=2
                                   ! 66d = 42h = 01000010b → 0:1:000010
```
Device

↓

```
SEND IDY                           ! set C0=IL-SRQ (1)
```
Controller

→

Abbreviations: Controller Output Message, Controller Input Message.

```
IL Scope: SEND IDY
Ctrl.Out.Msg | Ctrl.Inp.Msg | Comment
IDY    (600) | ISR    (700) | interrupt service request (C0=1)
```

↓

```
S = SPOLL(1)                       ! return DS (42h) & set IL-SRQ=0
DISP S                             ! LCD: "66"
```
Controller

↓

```
SEND IDY                           ! set C0=IL-SRQ (0)
```
Controller

→

Abbreviations: Controller Output Message, Controller Input Message.

```
IL Scope: SEND IDY
Ctrl.Out.Msg | Ctrl.Inp.Msg | Comment
IDY    (600) | IDY    (600) | identify (C0=0)
```

↓

```
S = SPOLL(1)                       ! return DS (42h)
DISP S                             ! LCD: "66"
```
Controller

71B IL Interface

71B IL Interface - Interrupt Bit Mask

7	6	5	4	3	2	1	0
128	64	32	16	8	4	2	1

7	Interface clear message received
6	Become a listener
5	Become the loop controller
4	Become the talker
3	Service request message received
2	Device clear message received
1	Trigger message received
0	DDL or DDT message received

71B IL Interface - Status Byte

7	6	5	4	3	2	1	0
128	64	32	16	8	4	2	1

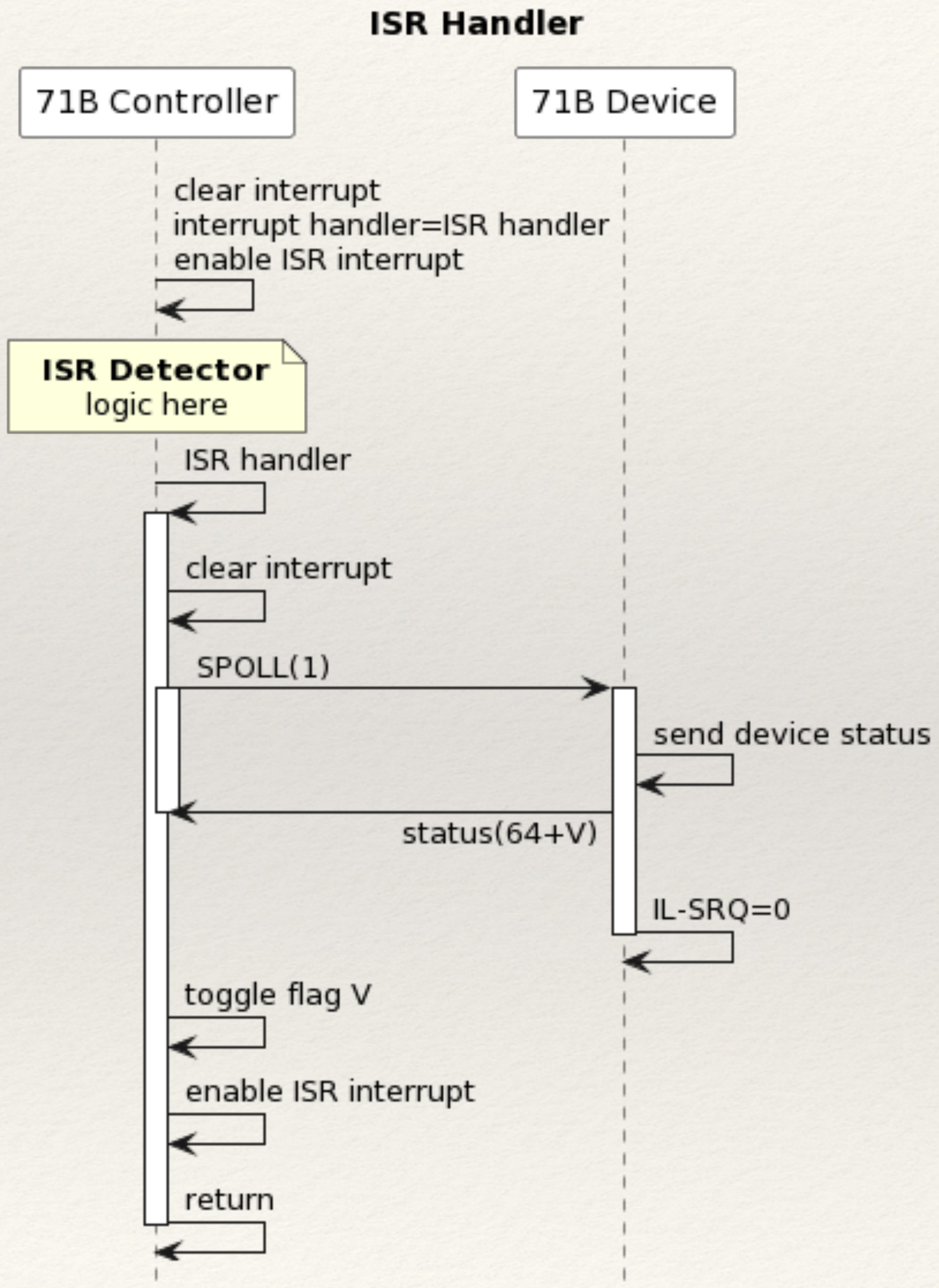
7	Data transfer occurring on the loop
6	I am a listener
5	I am the loop controller
4	I am the talker
3	Service request detected
2	Asynchronous request detected
1	Remote mode enabled
0	Local lockout enabled

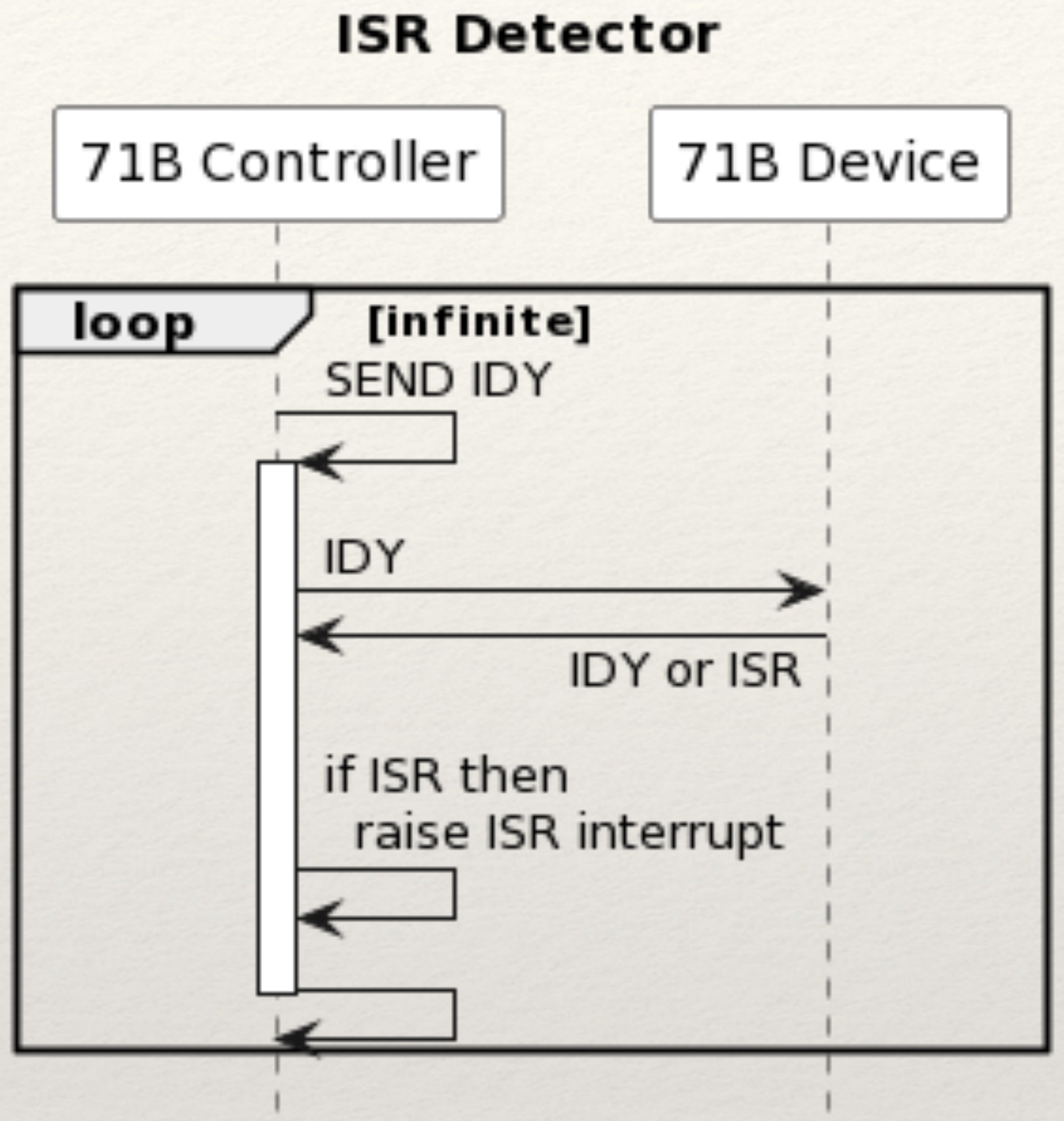
Unidirectional Communication

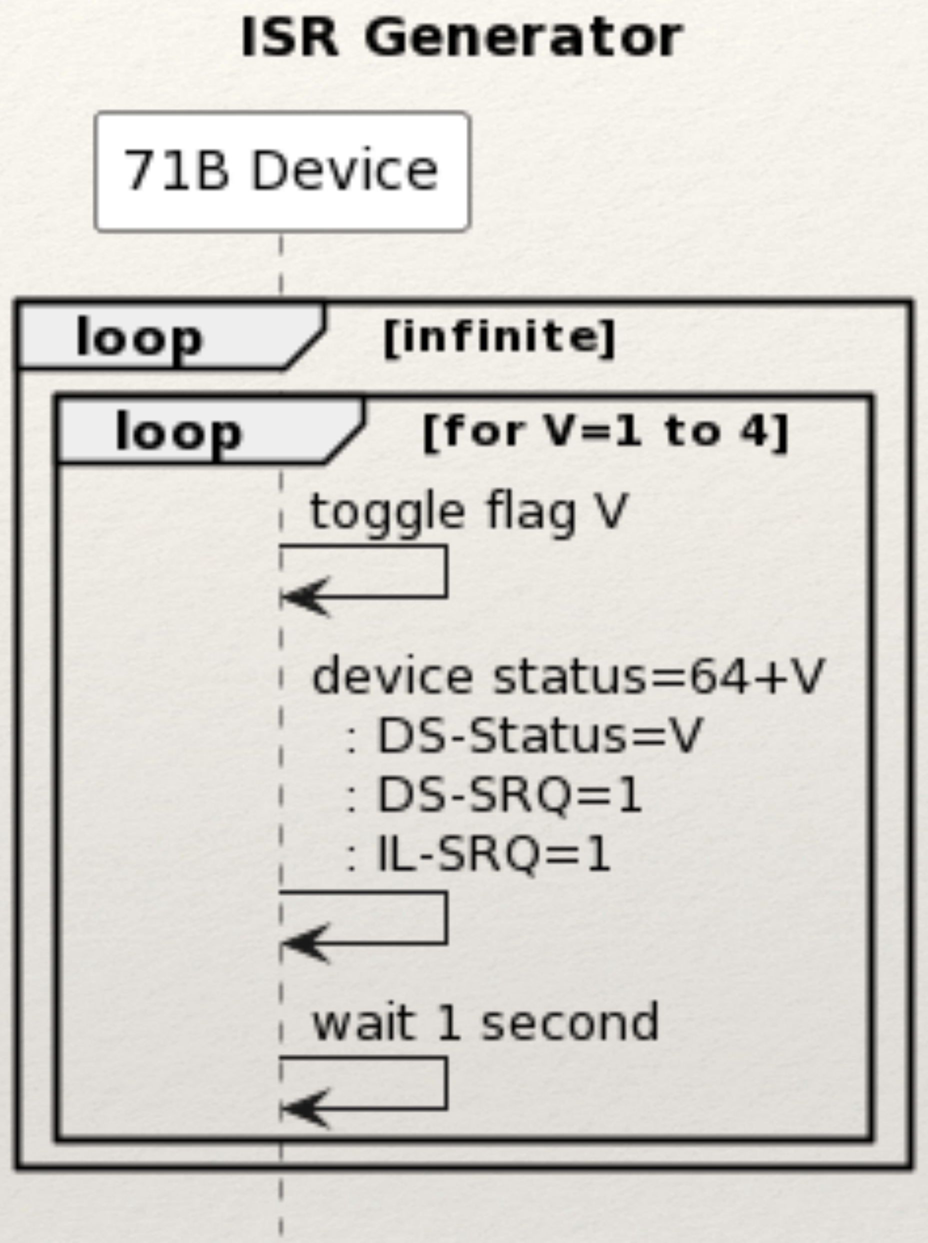
Demonstration Video

Unidirectional Communication

Using Interrupt Handler - UC1D & UC2C Diagrams







Unidirectional Communication

Using Interrupt Handler - UC1D & UC2C Implementation

Device

```
 5 ! UC1D              Unidirectional comm. prog. 1, device side
10 CFLAG 1,2,3,4       Clear flags

20 FOR V=1 TO 4        For-Loop, set or clear flags 1 to 4
25   IF FLAG(V)=0        Toggle flag using device status value
       THEN SFLAG V
       ELSE CFLAG V
30
     REQUEST 64+V        Raise SRQ bit while passing flag value
35   WAIT 1              Wait 1 second to let controller handle SRQ

40 NEXT V              For-Loop End
 @ GOTO 20             Infinite loop
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

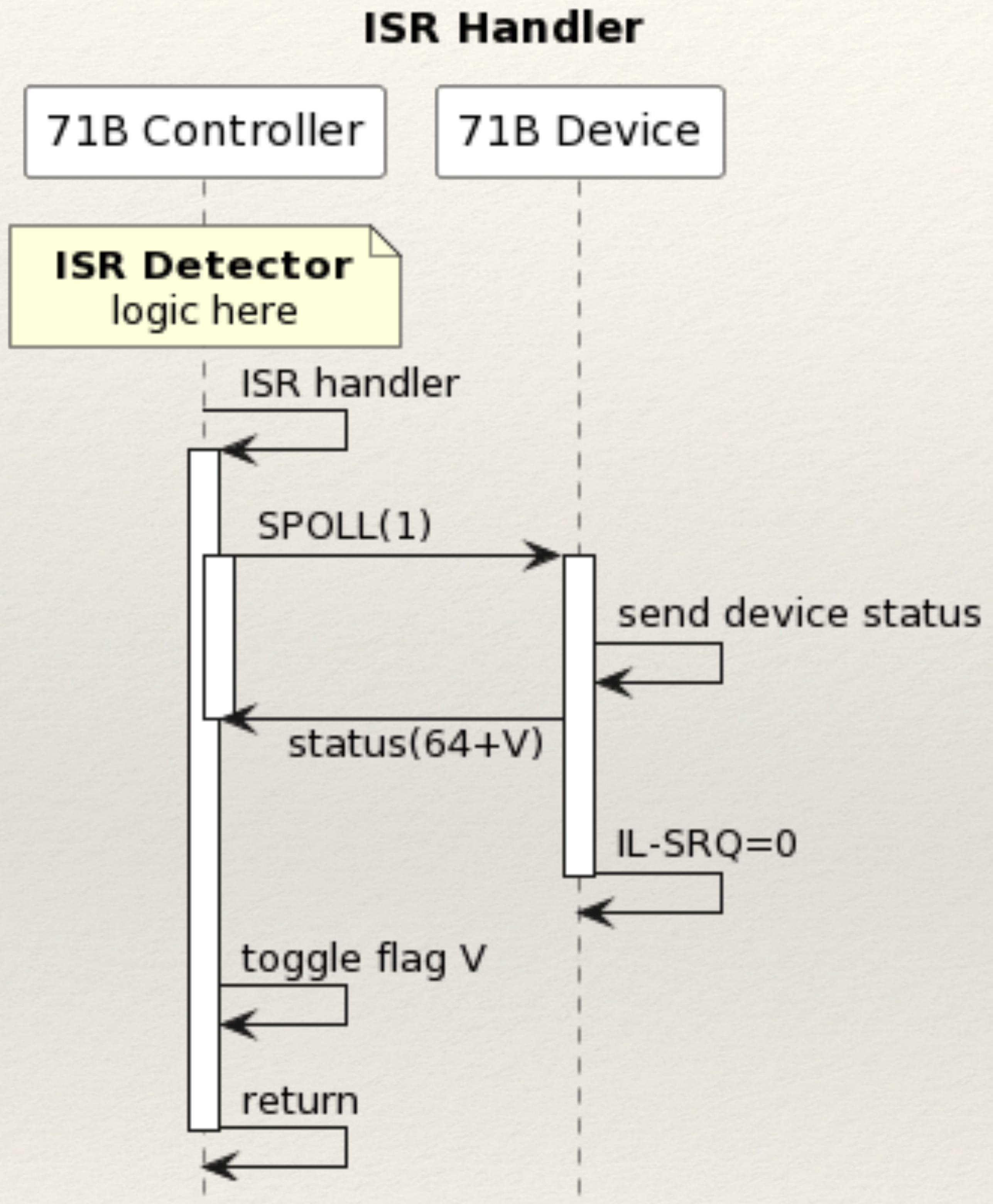
Controller

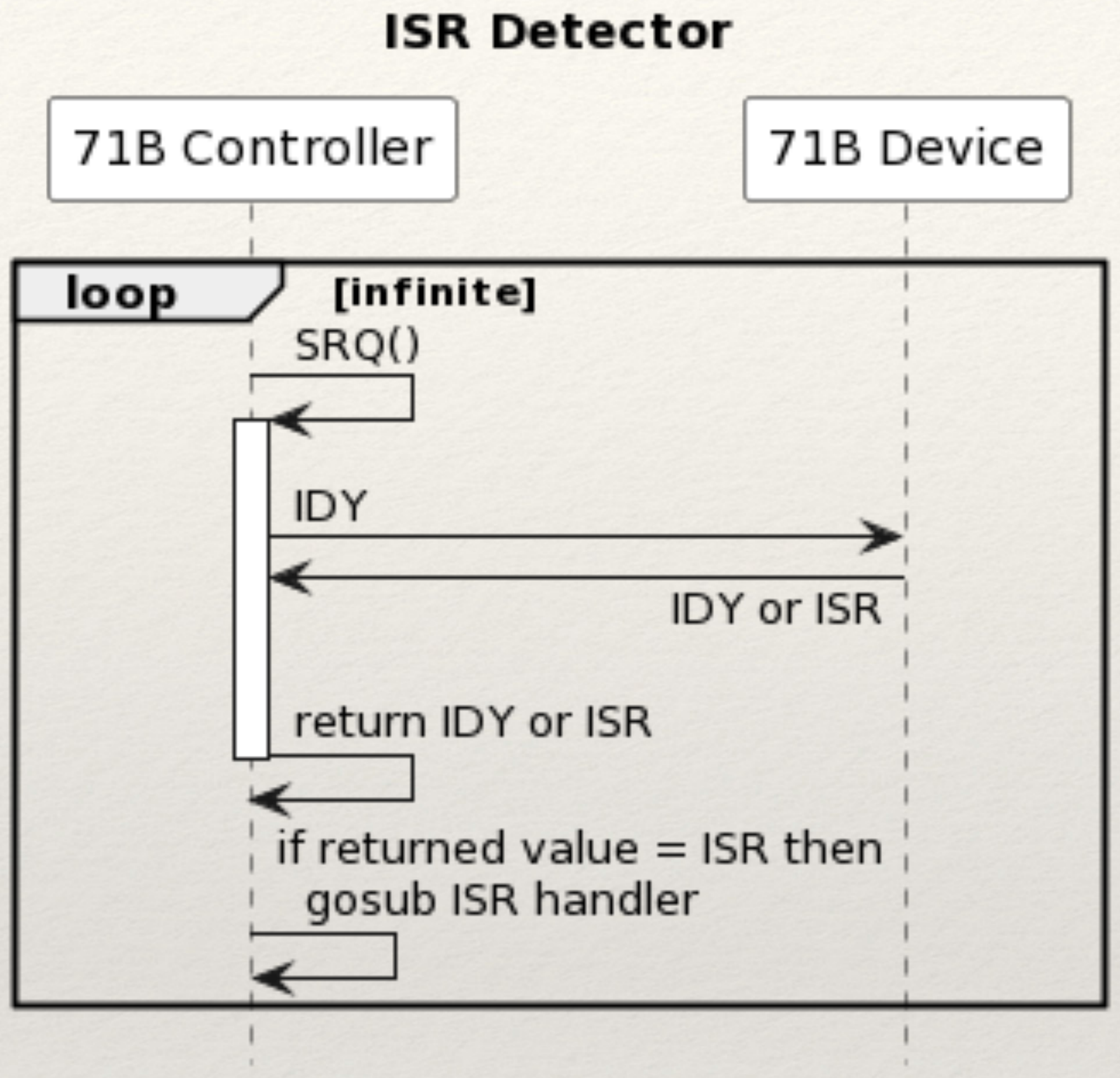
```
 5 ! UC2C              Unidirectional comm. prog. 2, controller side
10 CFLAG 0,1,2,3,4     Clear flags

15 CONTROL ON          Reconfigure loop

20 N=READINTR          Clear interrupt
 @ ON INTR GOSUB 35    Set interrupt handler to ISR handler
 @ ENABLE INTR 8       Enable interrupt on ISR [C0=1]

25 SEND IDY            If C0=1 on identify frame return then
                         Interrupt handler will be called
30 GOTO 25             Infinite poll loop

35 N=READINTR          Enter ISR handler, clear interrupt
 @ D=SPOLL(1)          Get device status
40 V=BINAND(D,63)      Extract value from device status
 @ IF FLAG(V)=0        Toggle flag using device status value
     THEN SFLAG V
     ELSE CFLAG V
45 ENABLE INTR 8       Re-enable service request interrupt

55 RETURN              Leave ISR handler
```

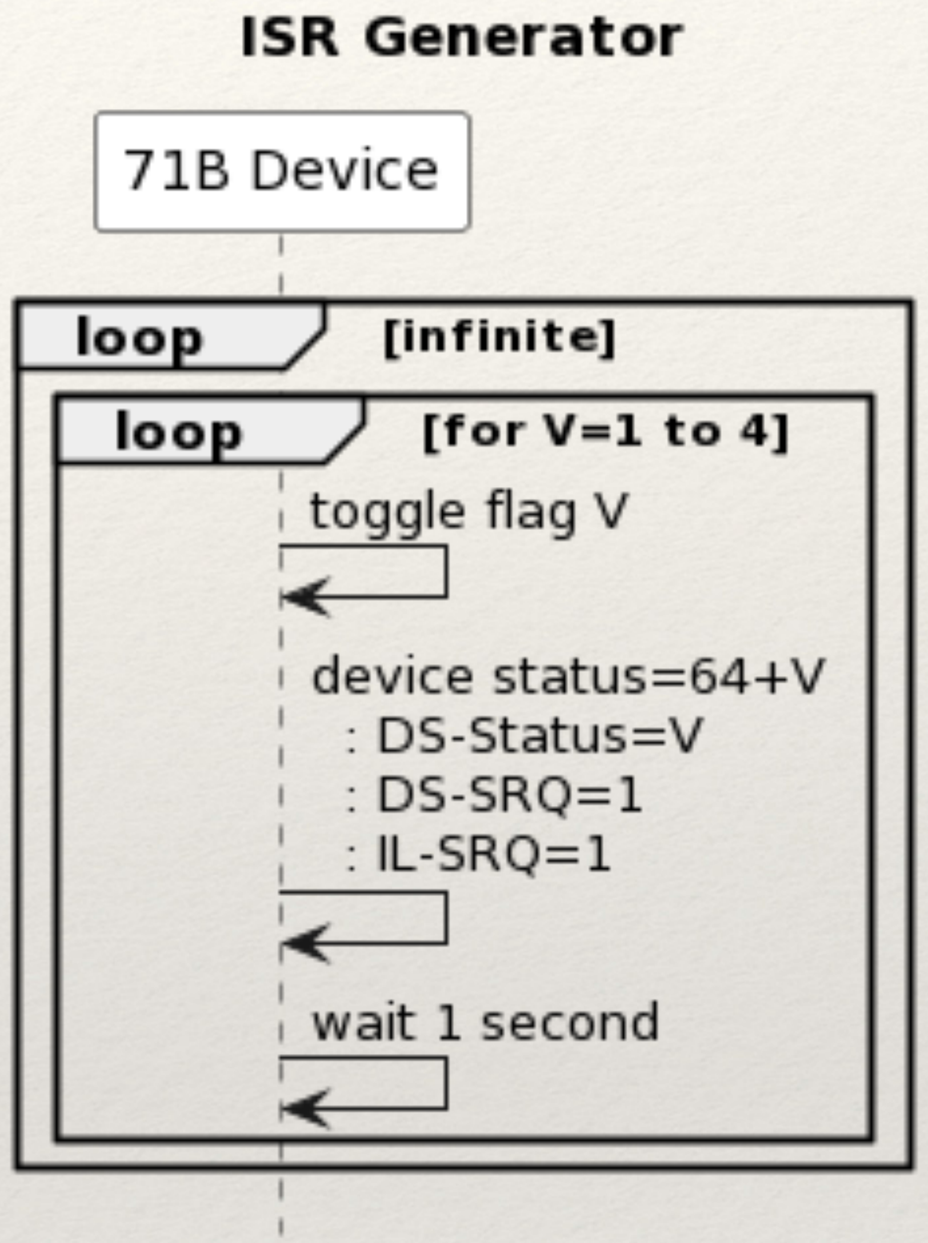
Default
Interrupt
Communication
Device Status
Device Address
Service Request

Unidirectional Communication

Using SRQ() Function - Diagrams







Unidirectional Communication

Using SRQ() Function - Implementation

Device

```
 5 ! UC1D                     Unidirectional comm. prog. 1, device side
10 CFLAG 1,2,3,4              Clear flags


20 FOR V=1 TO 4               For-Loop, set or clear flags 1 to 4
25   IF FLAG(V)=0              Toggle flag using device status value
       THEN SFLAG V
       ELSE CFLAG V
30
     REQUEST 64+V              Raise SRQ bit while passing flag value
35   WAIT 1                    Wait 1 second to let controller handle SRQ


40 NEXT V                     For-Loop End
 @ GOTO 20                    Infinite loop
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

Controller

```
 5 ! UC3C                     Unidirectional comm. prog. 3, controller side
10 CFLAG 0,1,2,3,4            Clear flags

15 CONTROL ON                 Reconfigure loop


25 IF SRQ()=1 THEN            If identify frame return 1 then
     GOSUB 35                  Call ISR handler
30 GOTO 25                    Infinite poll loop

35                            Enter ISR handler
   D=SPOLL(1)                 Get device status
40 V=BINAND(D,63)             Extract value from device status
 @ IF FLAG(V)=0               Toggle flag using device status value
     THEN SFLAG V
     ELSE CFLAG V


55 RETURN                     Leave ISR handler
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

Unidirectional Communication

Loop Latency Demonstration

Value :	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Flag 1:	→1	1	1	1	→1	1	1	1	→1	1	1	1	→1	1
Flag 2:	2	→2	2	2	2	→2	2	2	2	→2	2	2	2	→2
Flag 3:	3	3	→3	3	3	3	→3	3	3	3	→3	3	3	3
Flag 4:	4	4	4	→4	4	4	4	→4	4	4	4	→4	4	4

Loop Latency Test:

A. *load device application (UC1D) into a 71B that will act as a device*

B. *load controller application (UC2C or UC3C) into a 71B that will act as the controller*

C. *set wait value in device application (line 35) to .1 second (100 ms)*

D. *with HP-IL cables, connect two HP-71B together*

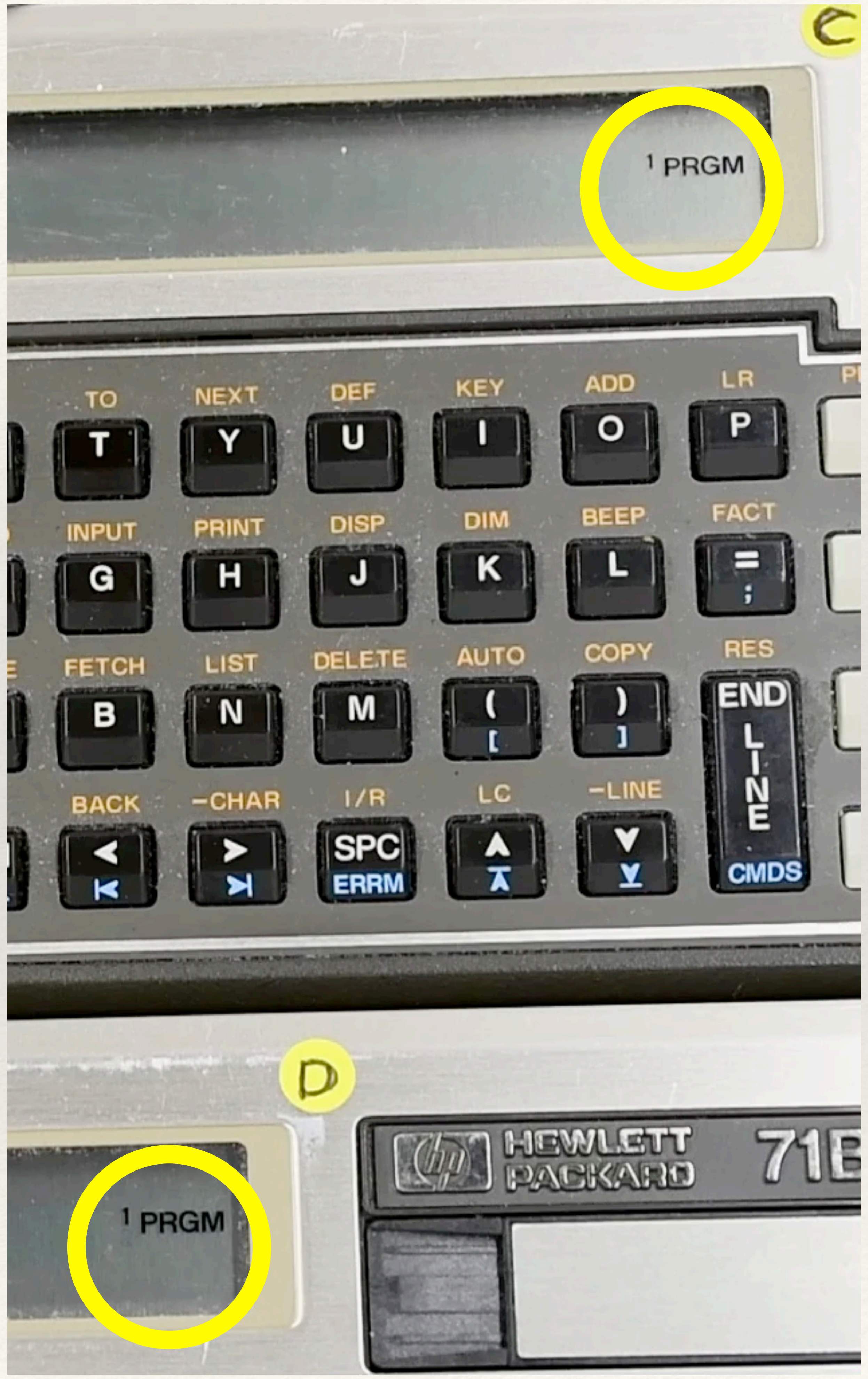
E. *start controller application then start device application*

F. *loop until flags value become desynchronized*

1. *stop device application then stop controller application*
2. *reduce wait value in device application (line 35) by 0.01 second (10 ms)*
3. *reset device status to 0 (REQUEST 0)*
4. *start controller application then start device application*

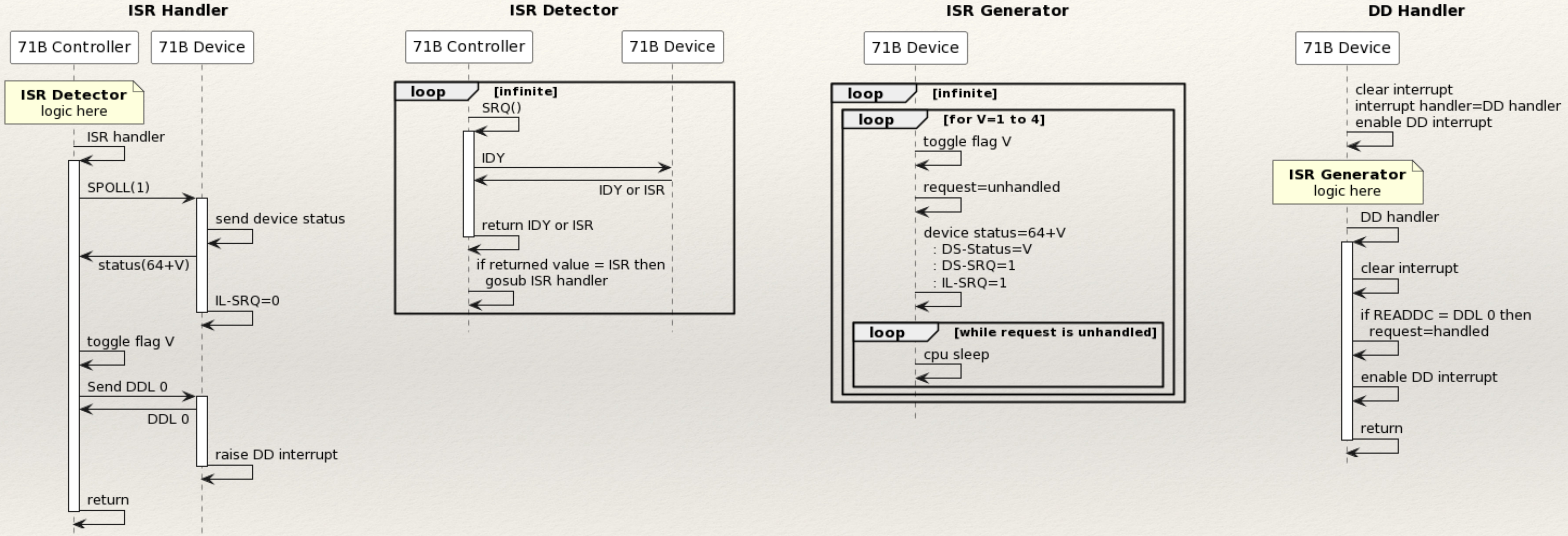
My experiments show that flags desynchronization appears near 55 ms for SSRIC (interrupt handler) and 12 ms for SSRSC (SRQ function).

With this configuration, this is the minimum latency between the 71B-device updating its device status and the 71B-Controller getting the information from the device.



Bidirectional Communication

Using SRQ() Function and Device Dependent Messages - BC1D & BC2C Diagrams



Bidirectional Communication

Using SRQ() Function and Device Dependent Messages - BC1D & BC2C Implementation

Device

```
 5 ! BC1D                       Bidirectional comm. prog. 1, device side
10 CFLAG 1,2,3,4                Clear flags
15 N=READINTR                   Clear interrupt
 @ ON INTR GOSUB 45             Set interrupt handler
 @ ENABLE INTR 1                Enable interrupt on device dependent messages
20 FOR V=1 TO 4                 For-Loop to set or clear flags 1 to 4
25   IF FLAG(V)=0                 Toggle flag using device status value
       THEN SFLAG V
       ELSE CFLAG V
30   F=1                          service request = unhandled
 @   REQUEST 64+V                 Raise SRQ bit while passing flag value
35   IF F=1 THEN                  Loop while service request = unhandled
       SLEEP                        Put CPU to sleep
 @     GOTO 35                      Loopback
40 NEXT V                       For-Loop End
 @ GOTO 20                      Infinite loop
45 N=READINTR                   Enter interrupt handler
 @ C=READDDC                    [00..31: DDT 0..31] & [32..63: DDL 0..31]
 @ IF C=32 THEN                 If DDL 0 received
     F=0                          service request = handled
50 ENABLE INTR 1                Re-enable interrupt
 @ RETURN                       Leave
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

Controller

```
 5 ! BC2C                       Bidirectional comm. prog. 2, controller side
10 CFLAG 0,1,2,3,4              Clear flags

15 CONTROL ON                   Reconfigure loop

25 IF SRQ()=1 THEN              If identify frame return 1 (ISR) then
     GOSUB 35                     Call SRQ handler
30 GOTO 25                      Infinite poll loop

35                              Enter SRQ handler
   D=SPOLL(1)                   Get device status
40 V=BINAND(D,63)               Extract value from device status
 @ IF FLAG(V)=0                 Toggle flag using device status value
     THEN SFLAG V
     ELSE CFLAG V

50 SEND UNT UNL LISTEN 1 DDL 0 UNL  Notify device → SRQ handled
55 RETURN                       Leave SRQ handler
```

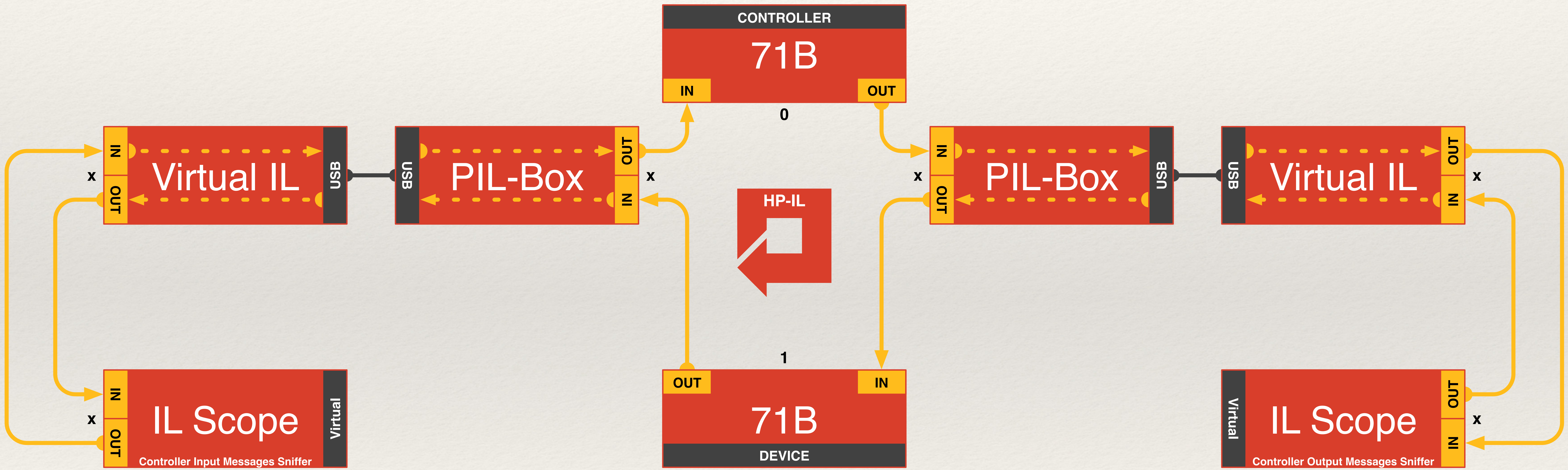
Default
Interrupt
Communication
Device Status
Device Address
Service Request

Communication II

One Controller
Multiple Devices

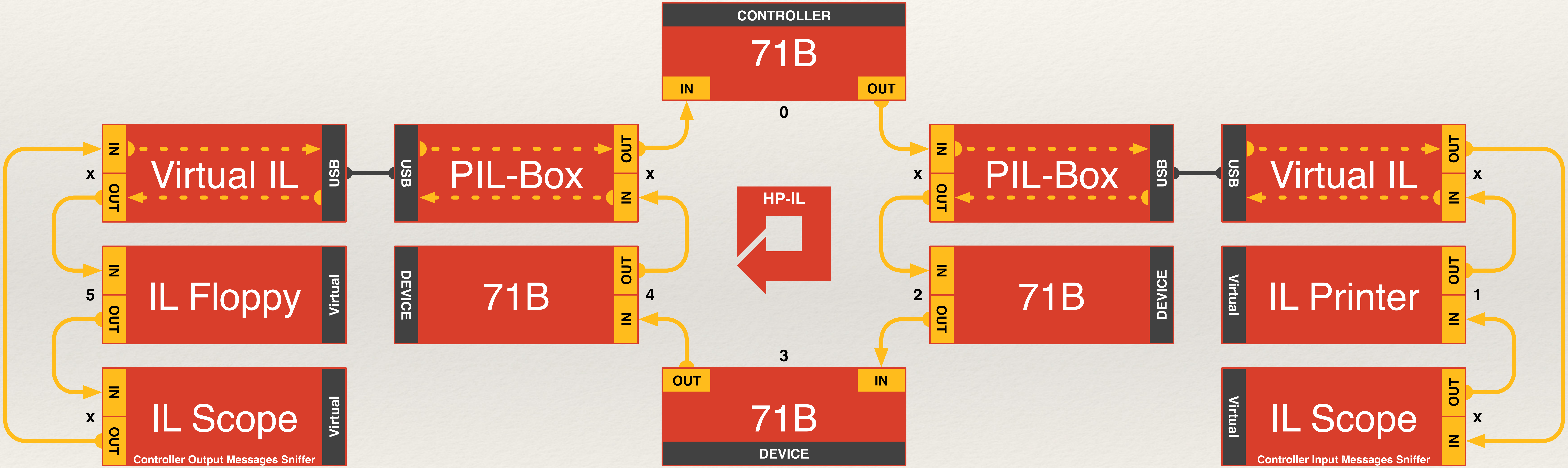
Loop Configuration 1

One controller with one device and two frames sniffers



Loop Configuration 2

One controller with five devices and two frames sniffers



Device Service Request

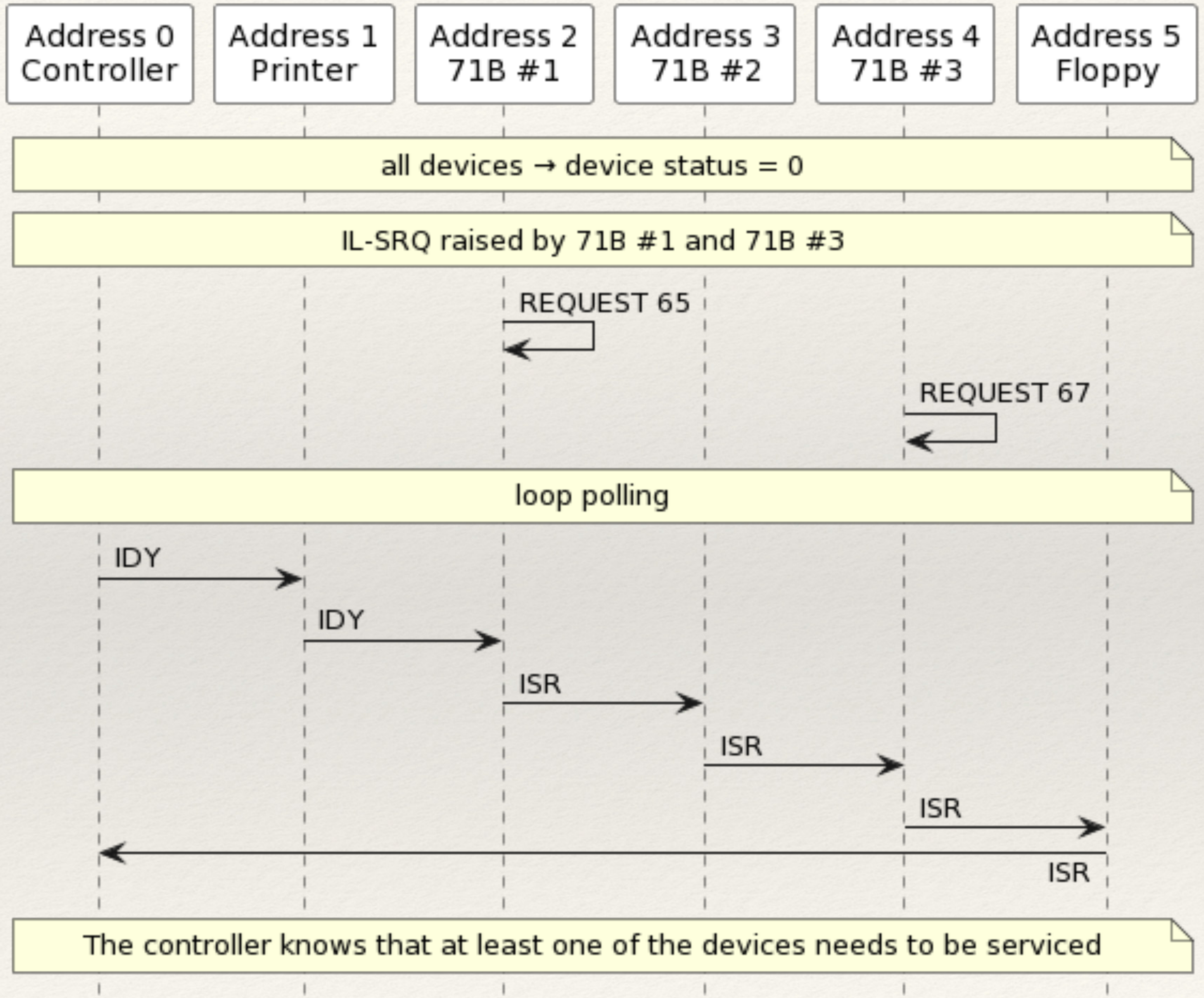
Multiple Devices

Controller
```
CONTROL ON           ! 71B controller → address = 0
                     ! Printer        → address = 1
                     ! 71B #1 device  → address = 2
                     ! 71B #2 device  → address = 3
                     ! 71B #3 device  → address = 4
                     ! 71B Floppy     → address = 5
```

71B #1 device
```
REQUEST 64+1         ! set IL-SRQ=1, DS-SRQ=1, DS-Sts=1
                     ! 65d = 41h = 01000001b → 0:1:000001
```

71B #3 device
```
REQUEST 64+3         ! set IL-SRQ=1, DS-SRQ=1, DS-Sts=3
                     ! 67d = 43h = 01000011b → 0:1:000011
```

Controller
```
SEND IDY             ! poll loop → service request needed
```



Serial Polling

Multiple Devices

Controller

```
S = SPOLL(1)                    ! get status from Printer
DISP S                          ! LCD: "0"
```

Controller

```
S = SPOLL(2)                    ! get status from 71B #1
DISP S                          ! LCD: "65"
```

Controller

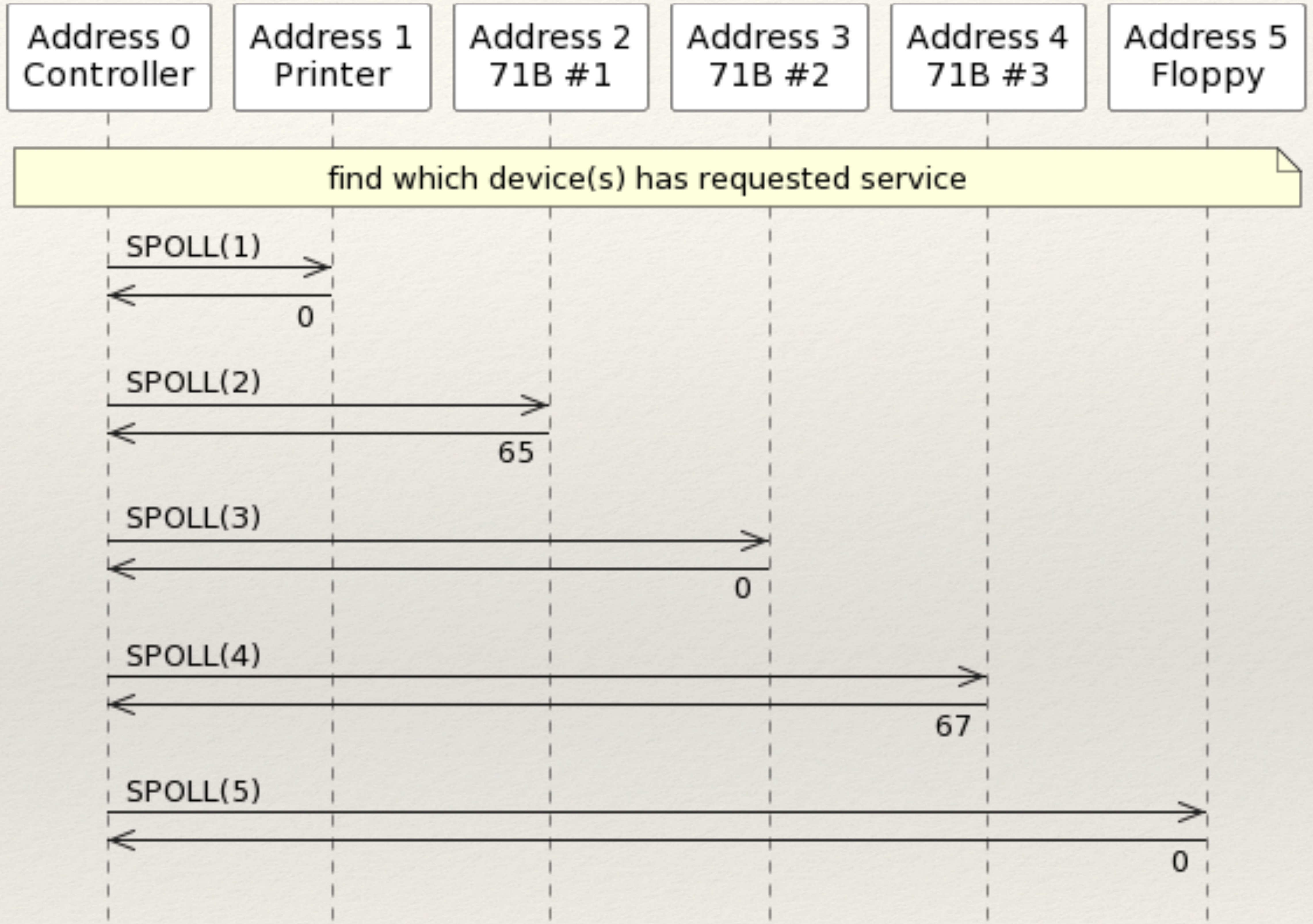
```
S = SPOLL(3)                    ! get status from 71B #2
DISP S                          ! LCD: "0"
```

Controller

```
S = SPOLL(4)                    ! get status from 71B #3
DISP S                          ! LCD: "67"
```

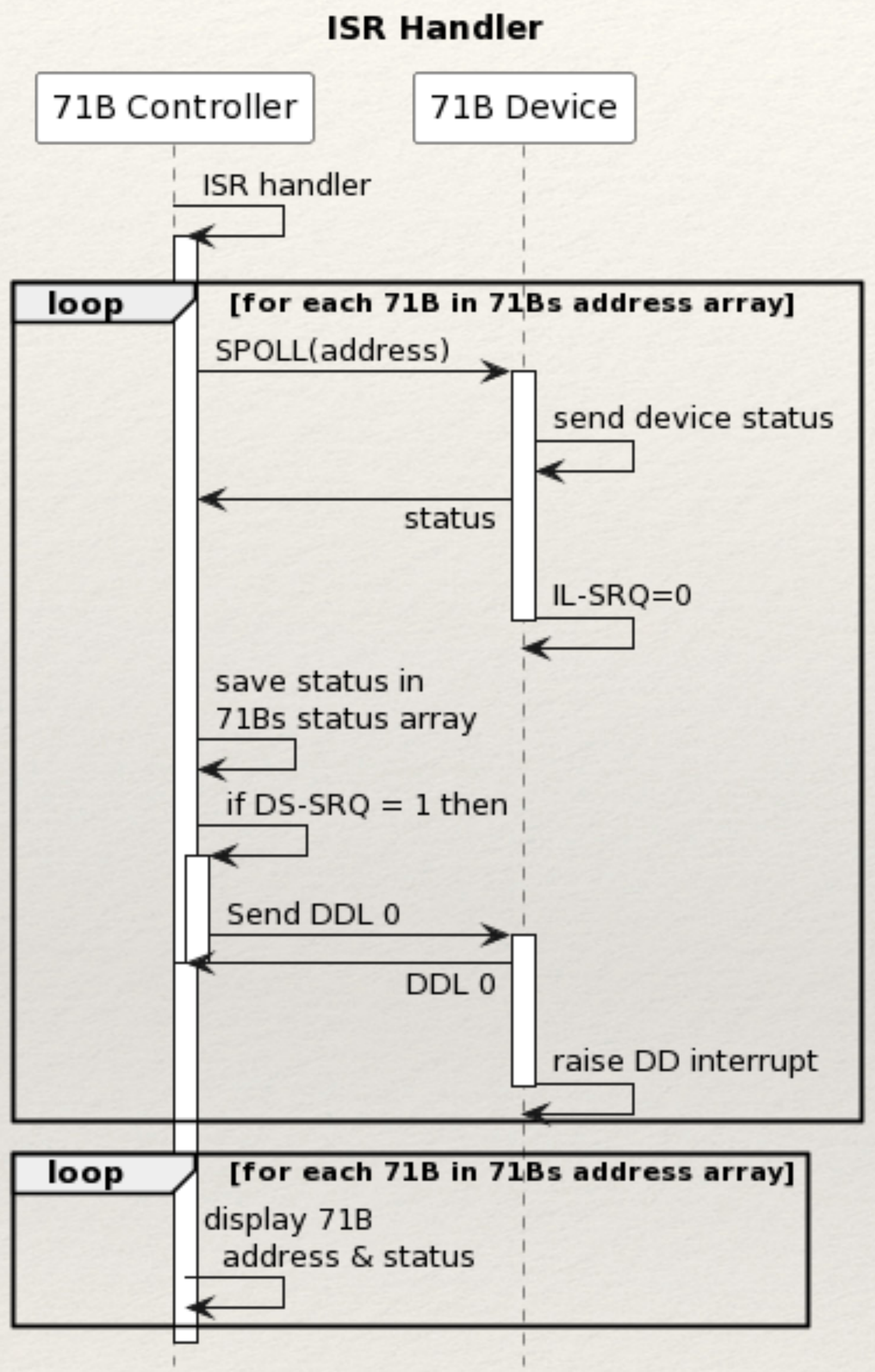
Controller

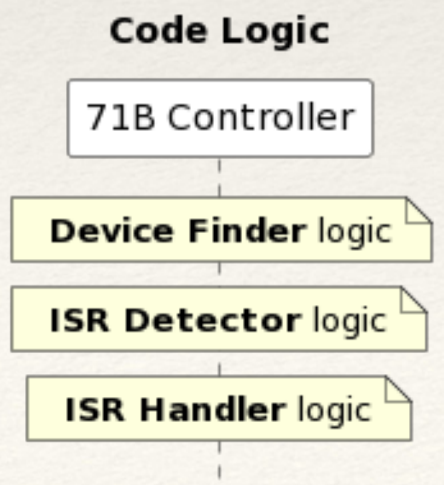
```
S = SPOLL(5)                    ! get status from Floppy
DISP S                          ! LCD: "0"
```

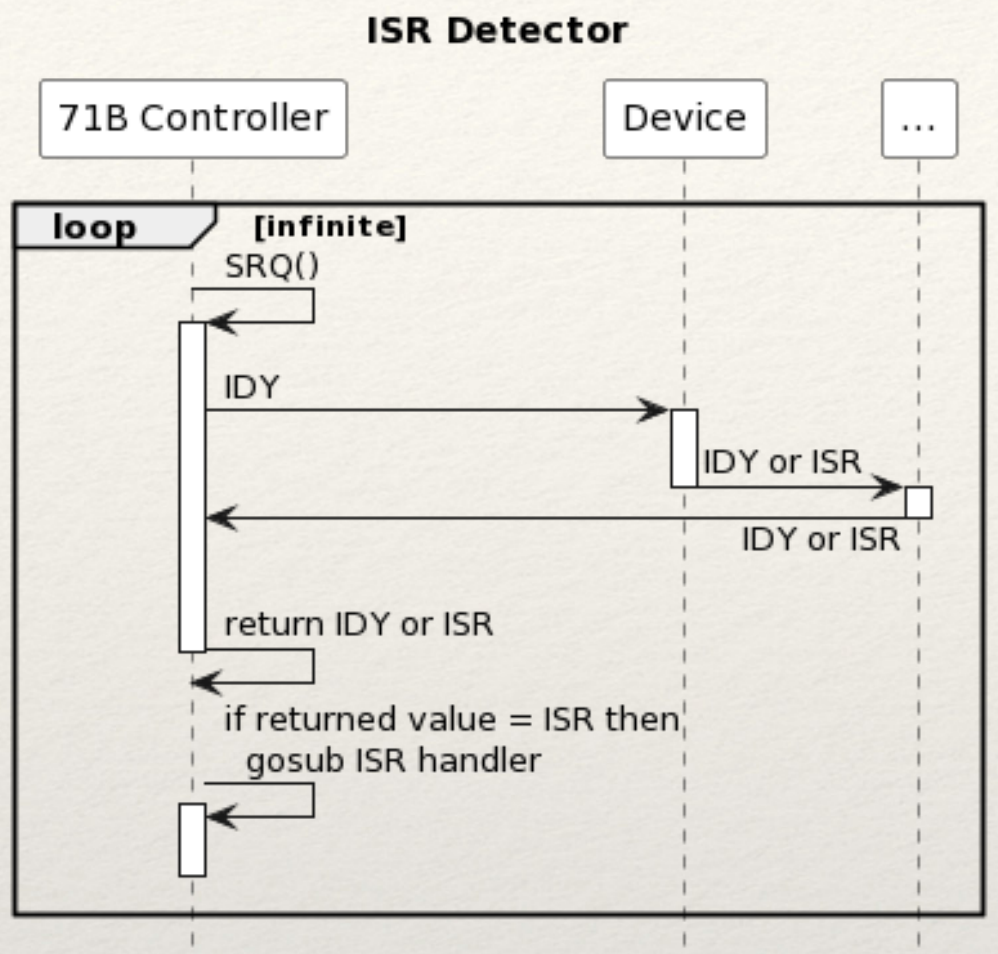


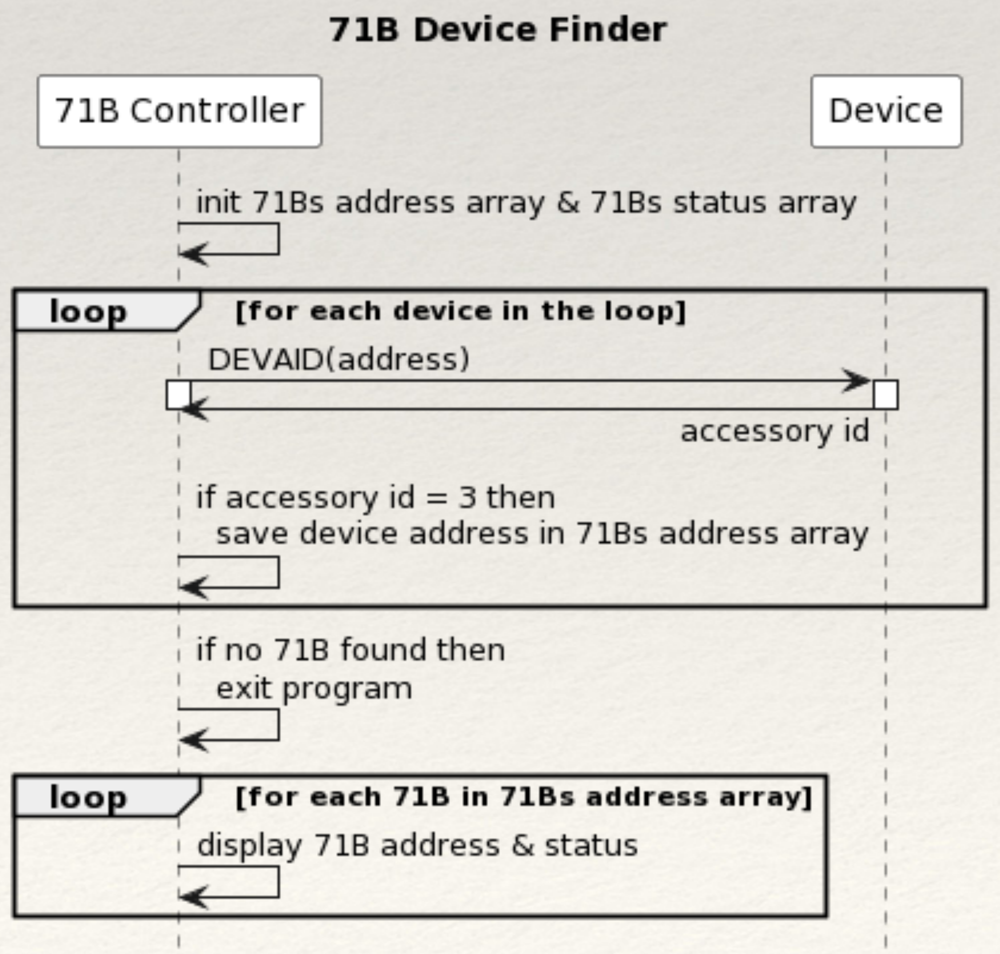
Bidirectional Communication

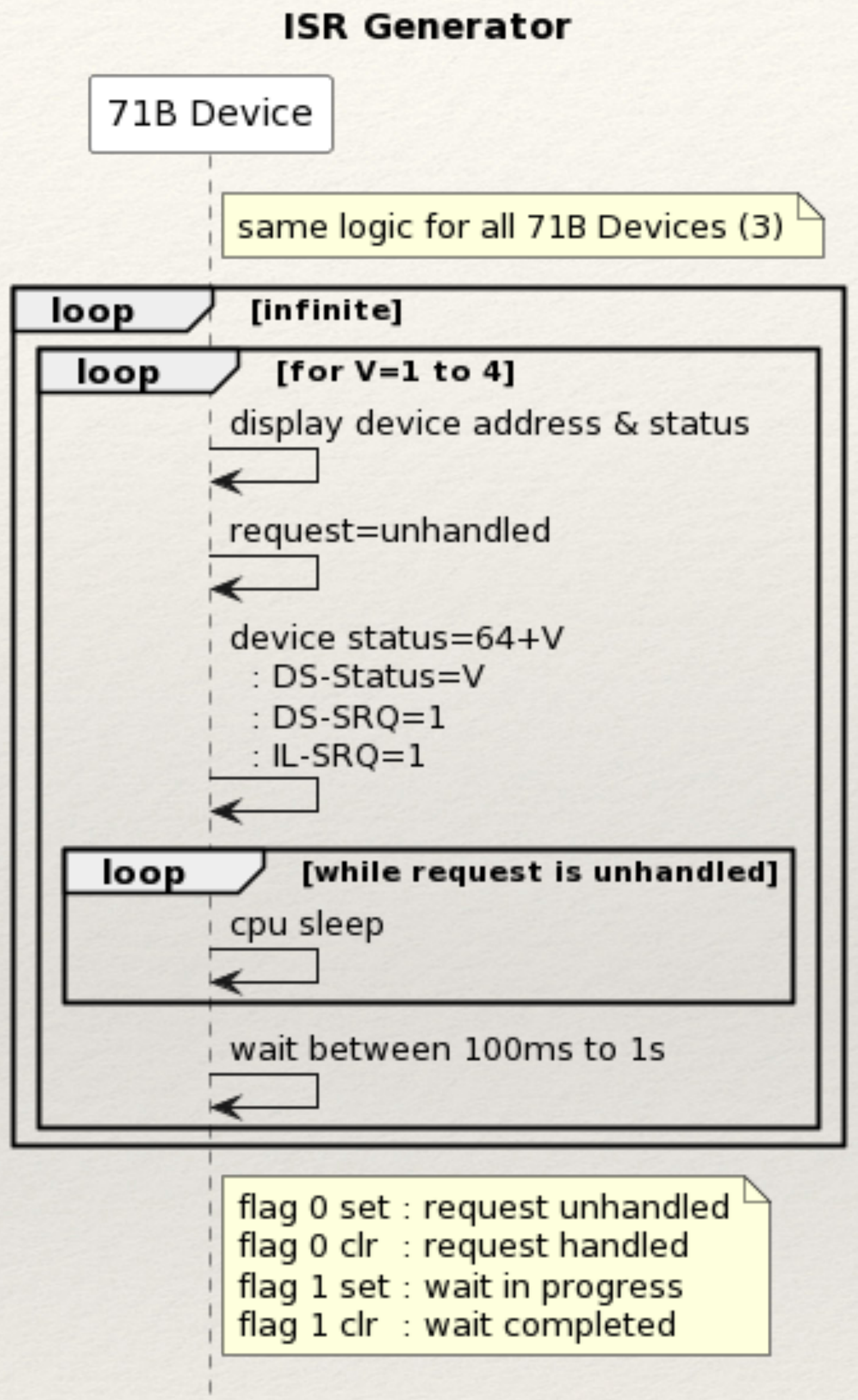
Multiple Devices - Serial Polling - BC3D & BC4C Diagrams

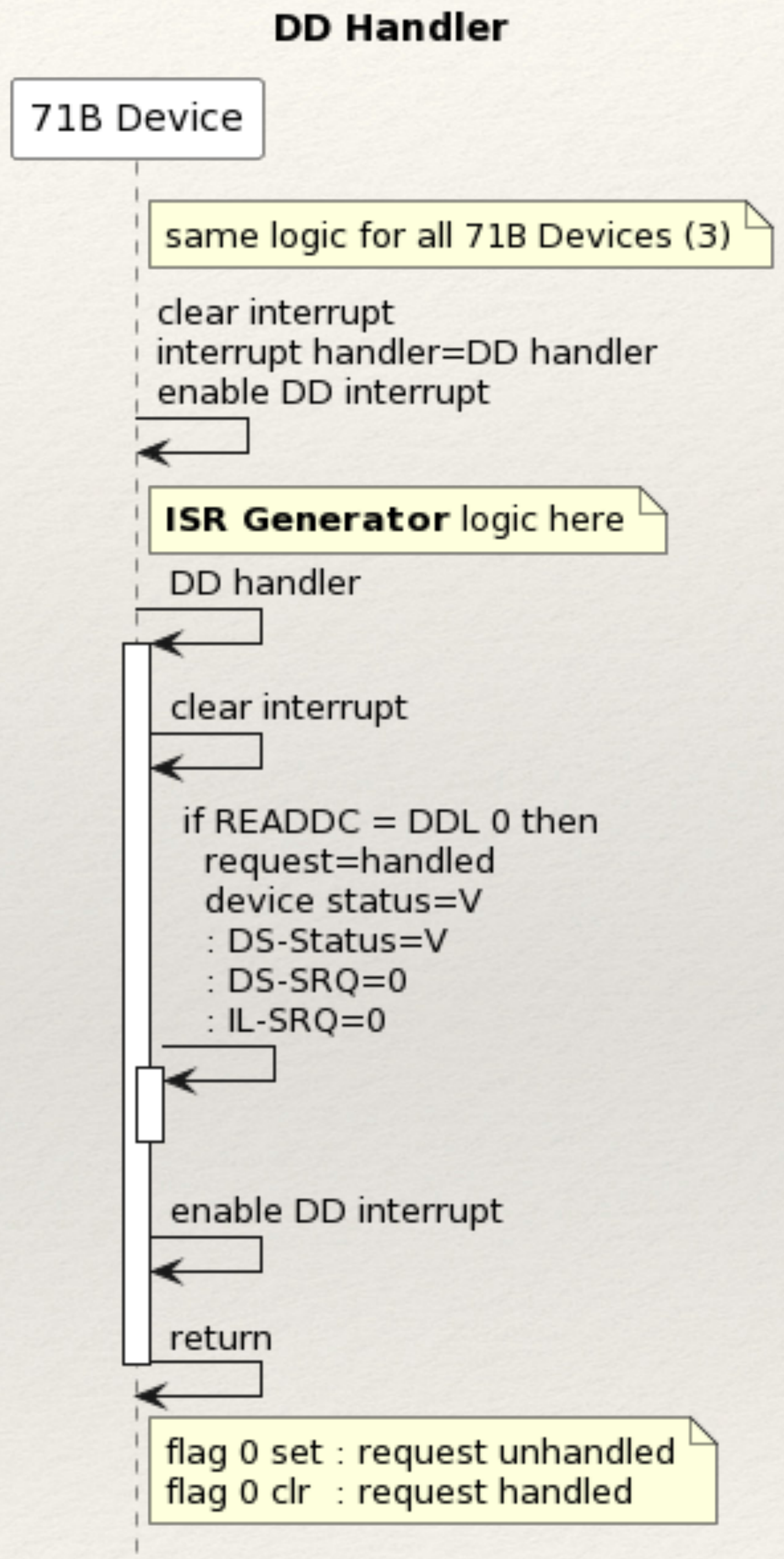












Bidirectional Communication

Multiple Devices - Serial Polling - BC3D Implementation

Device

```
 5 ! BC3D                              Bidirectional comm. prog. 3, device side
10 DELAY 0,0                           Set display as fast as possible
 @ RANDOMIZE                           Random number generator
 @ A=MYADDR                            Save my loop address
 @ CFLAG 0,1                           Flag 0 (waiting for DDL 0) & flag 1 (waiting)

15 N=READINTR                          Clear interrupt
 @ ON INTR GOSUB 50                    Set interrupt handler
 @ ENABLE INTR 1                       Enable interrupt on device dependent message

20 FOR V=1 TO 4                        Loop to set or clear flags 1 to 4
25   DISP "D="&STR$(A)&"  S="&STR$(V)  Ex.: "D=3  S=2" (Device & Status)
30   SFLAG 0                           Activate wait for DDL 0 message flag
 @   REQUEST 64+V                      Raise SRQ bit while passing flag value
35   IF FLAG(0)=1 THEN                 DDL 0 received ?
       SLEEP                           No: go to sleep
 @     GOTO 35                         Infinite DDL 0 wait loop
40   SFLAG 1                           Activate wait flag
 @   WAIT IP(RND*10+1)/10              Wait 100 ms to 1 sec, asynchronous simulation
 @   CFLAG 1                           Deactivate wait flag
45 NEXT V
 @ GOTO 20                             Infinite loop
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

Device

```
50 N=READINTR                          Enter interrupt handler, clear interrupt
 @ C=READDDC                           [00..31: DDT 0..31] & [32..63: DDL 0..31]
55 IF C=32 THEN                        If DDL 0 received then
     CFLAG 0                             Clear DDL 0 wait flag
 @   REQUEST V                           Clear SRQ bit
60 ENABLE INTR 1                       Re-enable device dependent interrupt
 @ RETURN                              Leave interrupt handler
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

Bidirectional Communication

Multiple Devices - Serial Polling - BC4C Implementation

Controller

```
 5 ! BC4C                          Bidirectional comm. prog. 4, controller side
10 OPTION BASE 0                   Make array index start at zero
 @ DELAY 0,0                       Set display as fast as possible
 @ DIM D$[60]                      Space for 8 devices (6 chr/device + some room)
15 CONTROL ON                      Reconfigure loop
 @ N=NLOOP()                       Get the number of devices in the loop
20 DESTROY A1, A2                  Delete devices address & devices status arrays
 @ SHORT A1(N)                     Create an address array for all devices
 @ A1(0)=0                         Initialize 71B found counter
25 FOR I=1 TO N                    Scan loop for 71Bs
30   IF DEVAID(I)=3 THEN           Is device a 71B ?
       A1(0)=A1(0)+1               Yes, increment 71B found counter
 @     A1(A1(0))=I                 Save 71b address in devices address array
35 NEXT I
40 IF A1(0)=0 THEN                 Quit if no 71B has been found
     DISP "No 71B found"
 @   DESTROY A1, A2                Exit: delete arrays
 @   END                           End of execution ...
45 SHORT A1(A1(0)),A2(A1(0))       Resize/create arrays to match 71Bs found

50 GOSUB 85                        Display what we have
```

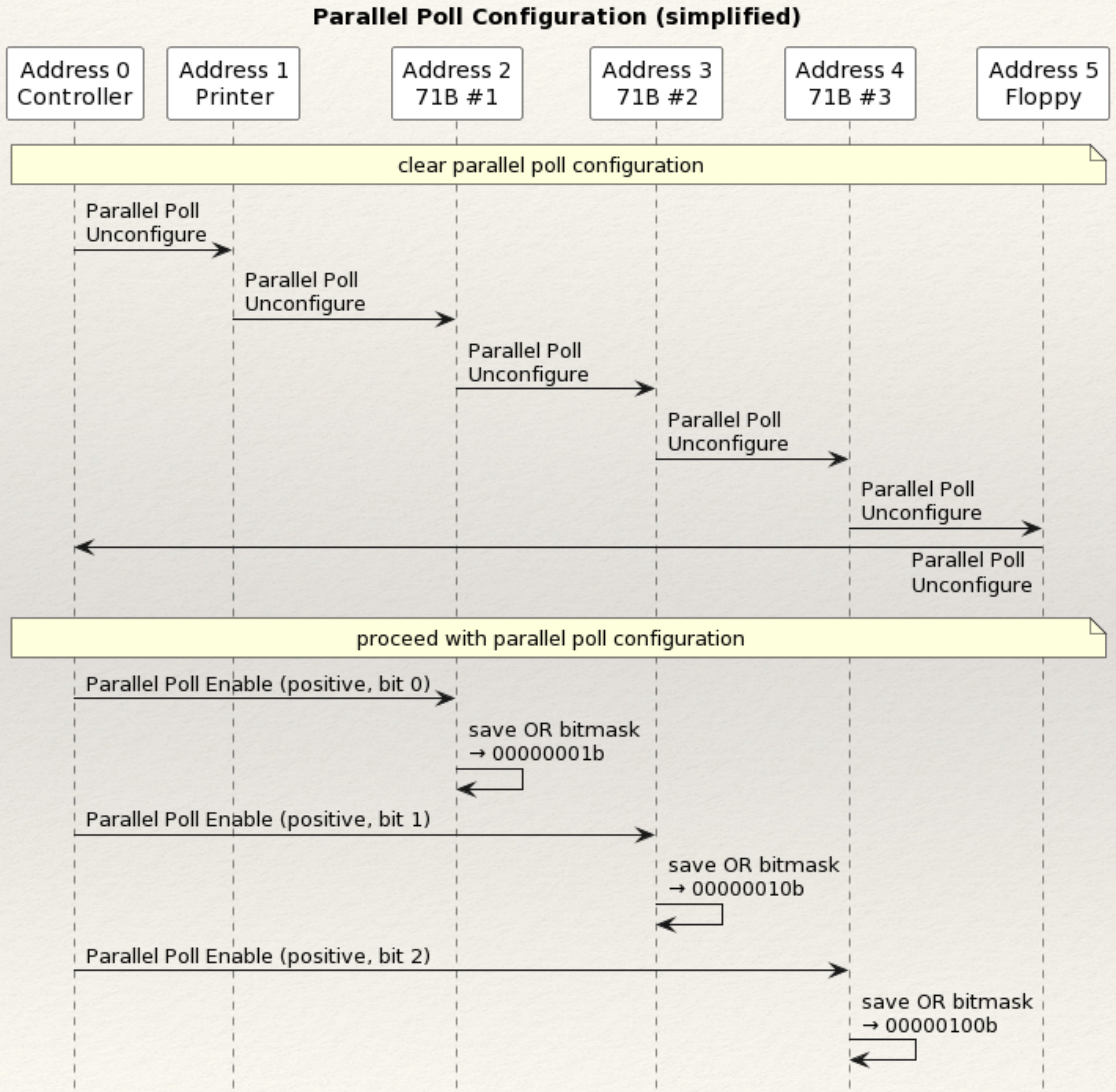
Default
Interrupt
Communication
Device Status
Device Address
Service Request

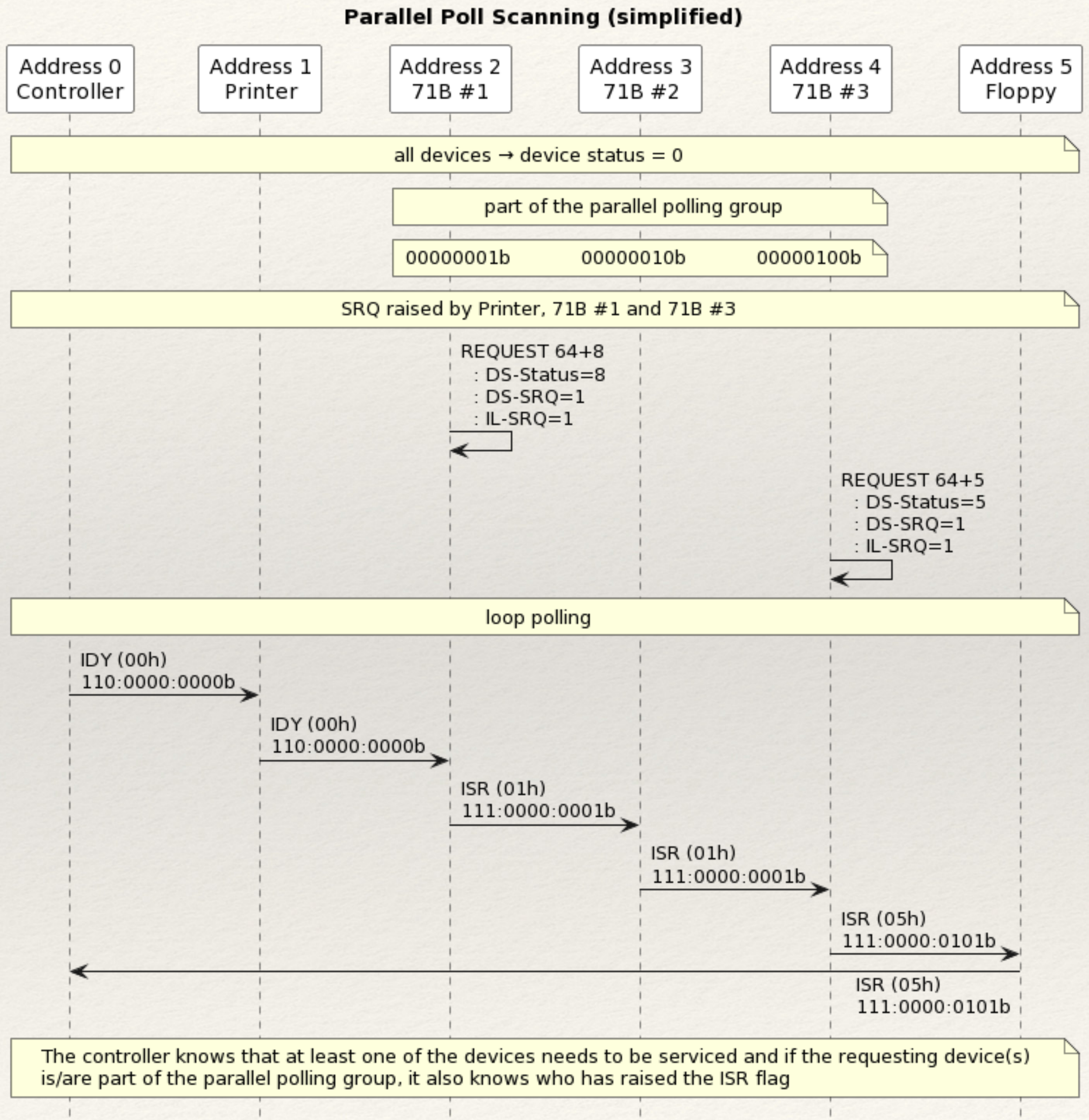
Controller

```
60 S=SRQ()                         Poll the loop to see If we have a service request
 @ IF S#0 THEN                     Do we have a service request ?
     GOSUB 70                      Yes: call SRQ handler
65
   GOTO 60                         Infinite loop polling
70 FOR I=1 TO A1(0)                Enter SRQ handler, for each 71B found
75
       A2(I)=SPOLL(A1(I))          Get 71B status and save it in status array
 @     IF BIT(A2(I),6)=1 THEN      Is SRQ bit set ?
         SEND UNT UNL LISTEN A1(I) DDL 0 UNL
80 NEXT I                          ↑ Yes, notify device → SRQ handled
85 D$=STR$(A1(0))                  Start string with the number of 71B found
90 FOR I=1 TO A1(0)                For each 71B found
 @   D$=D$&CHR$(92)&STR$(A1(I))&":"&DTH$(A2(I))[4,5]
 @ NEXT I                          ↑ Add 71B address then add 71B status
95 DISP D$                         Ex.: 3\2:41\3:80\4:00
 @ RETURN                          Leave SRQ handler,
                                   ↑ end sub for line 50 and 60
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

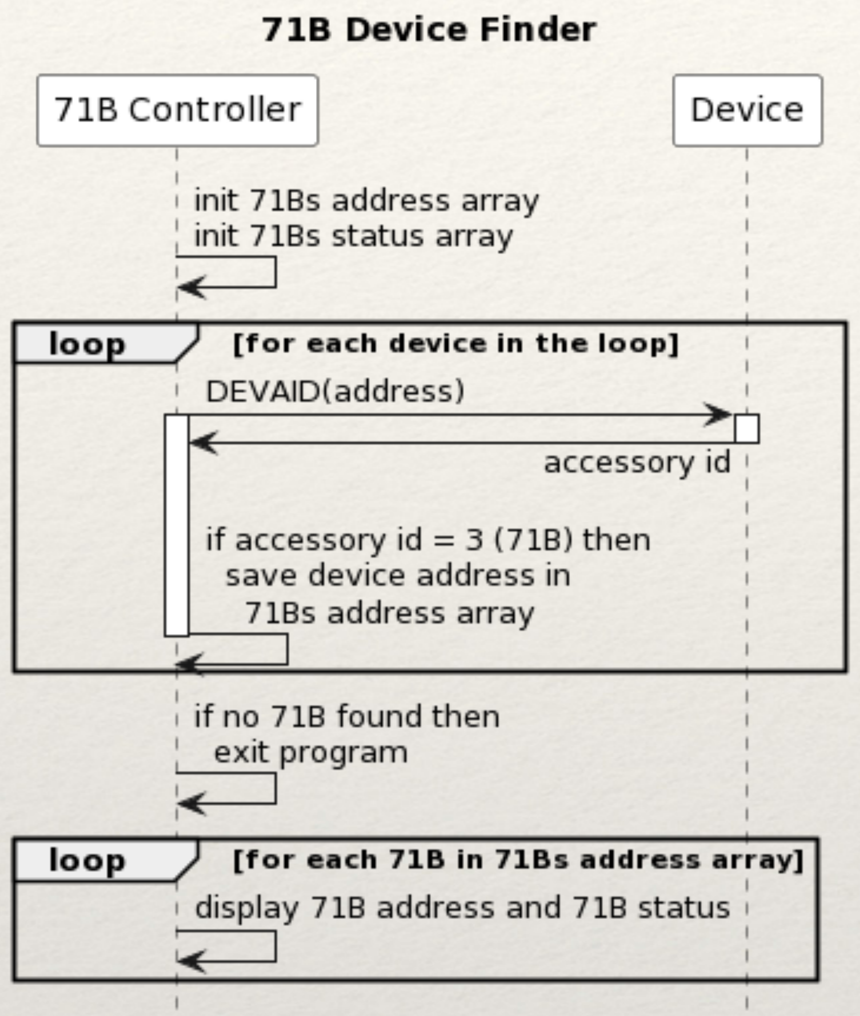
Parallel Polling

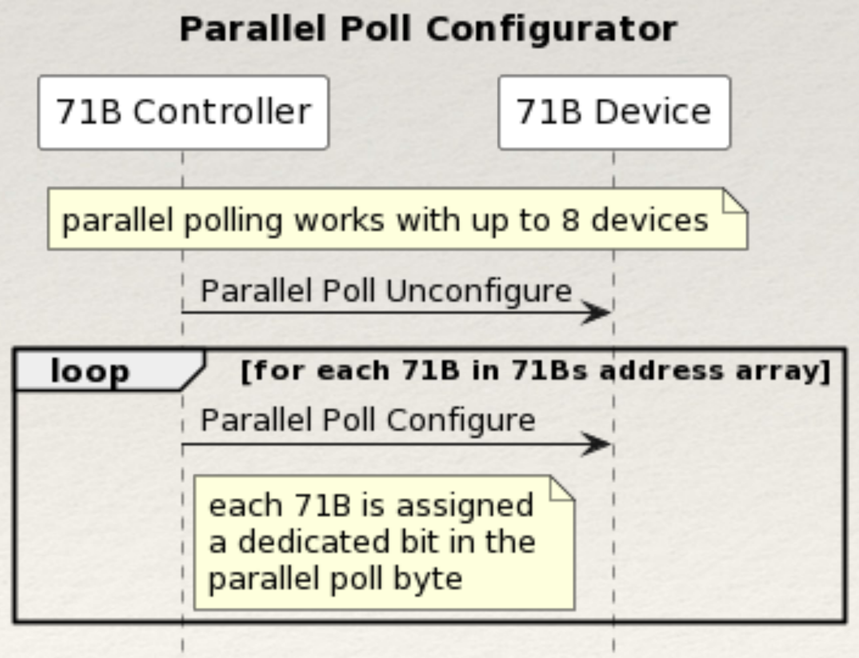


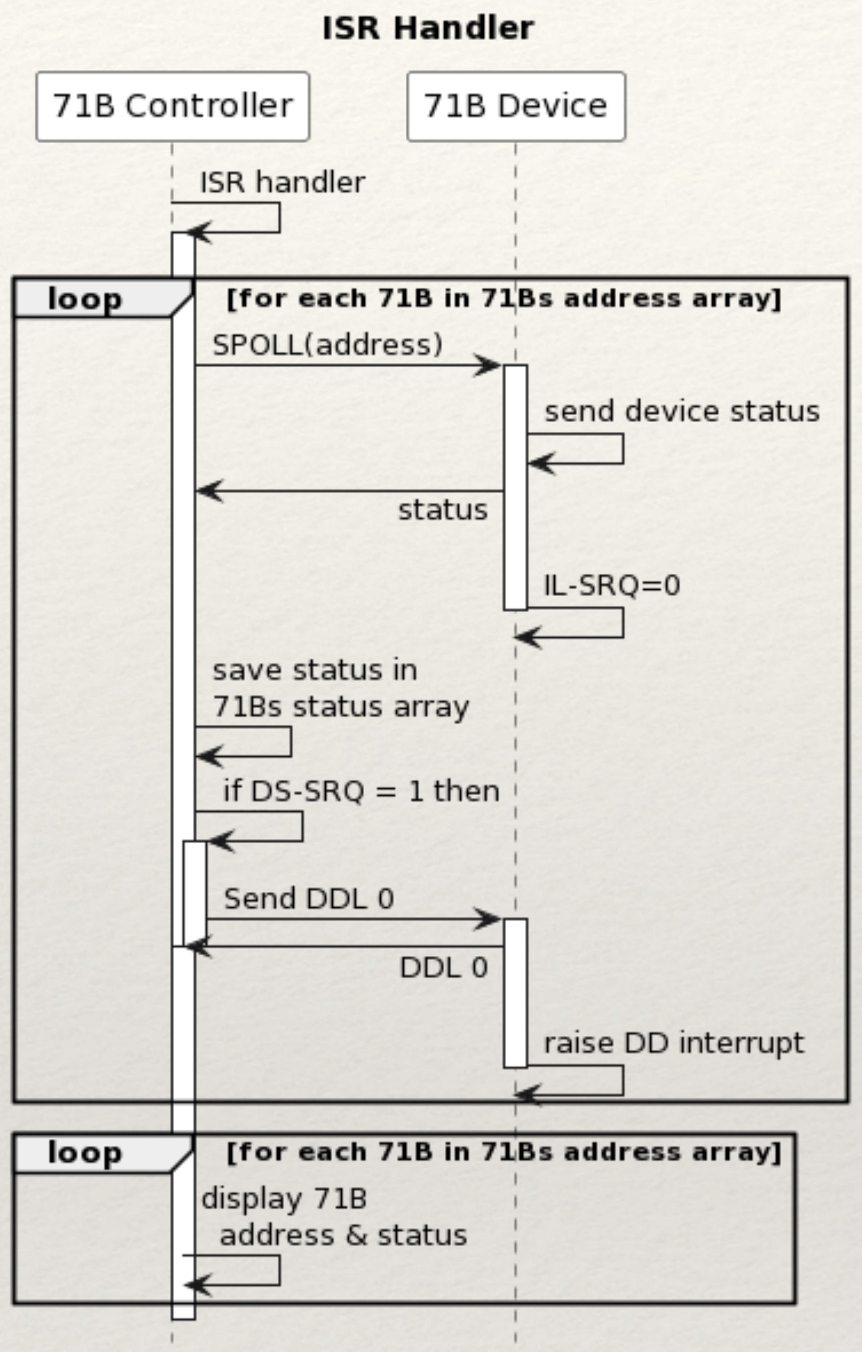


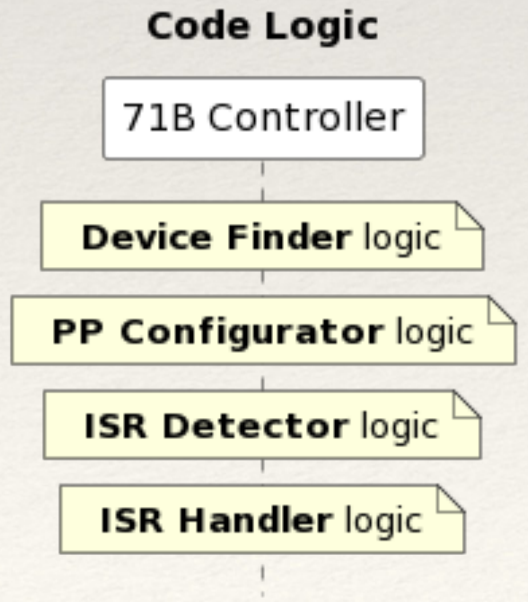
Bidirectional Communication

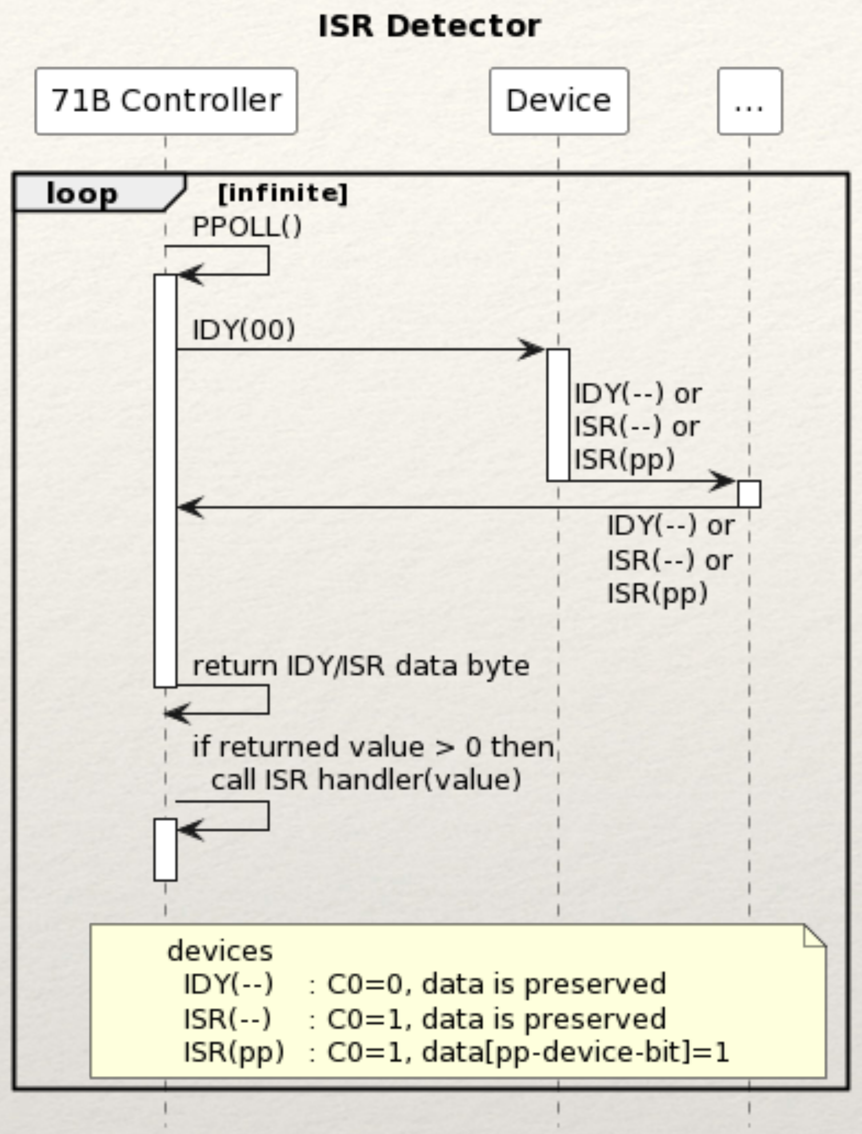
Multiple Devices - Parallel Polling - BC3D & BC5C Diagrams

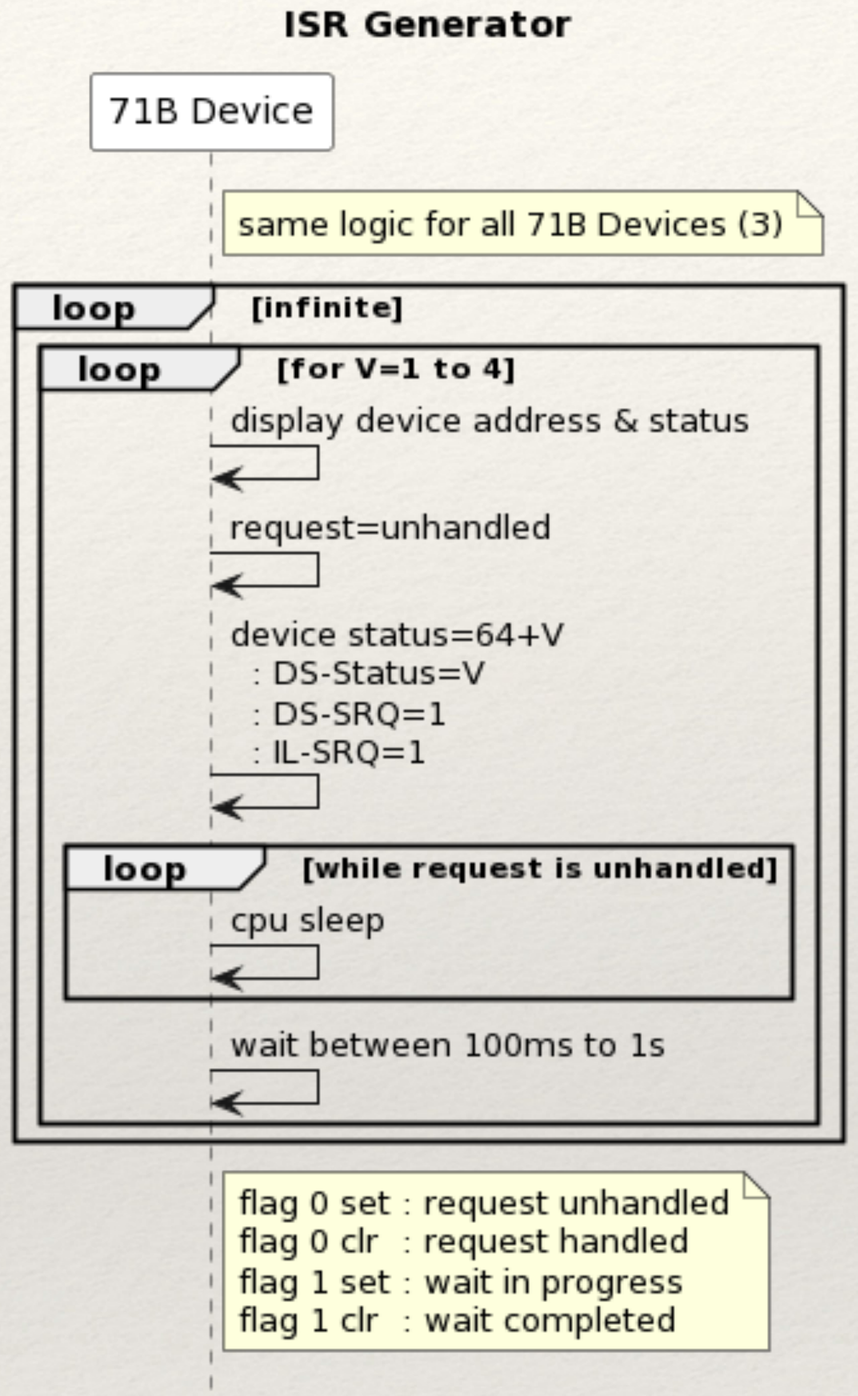


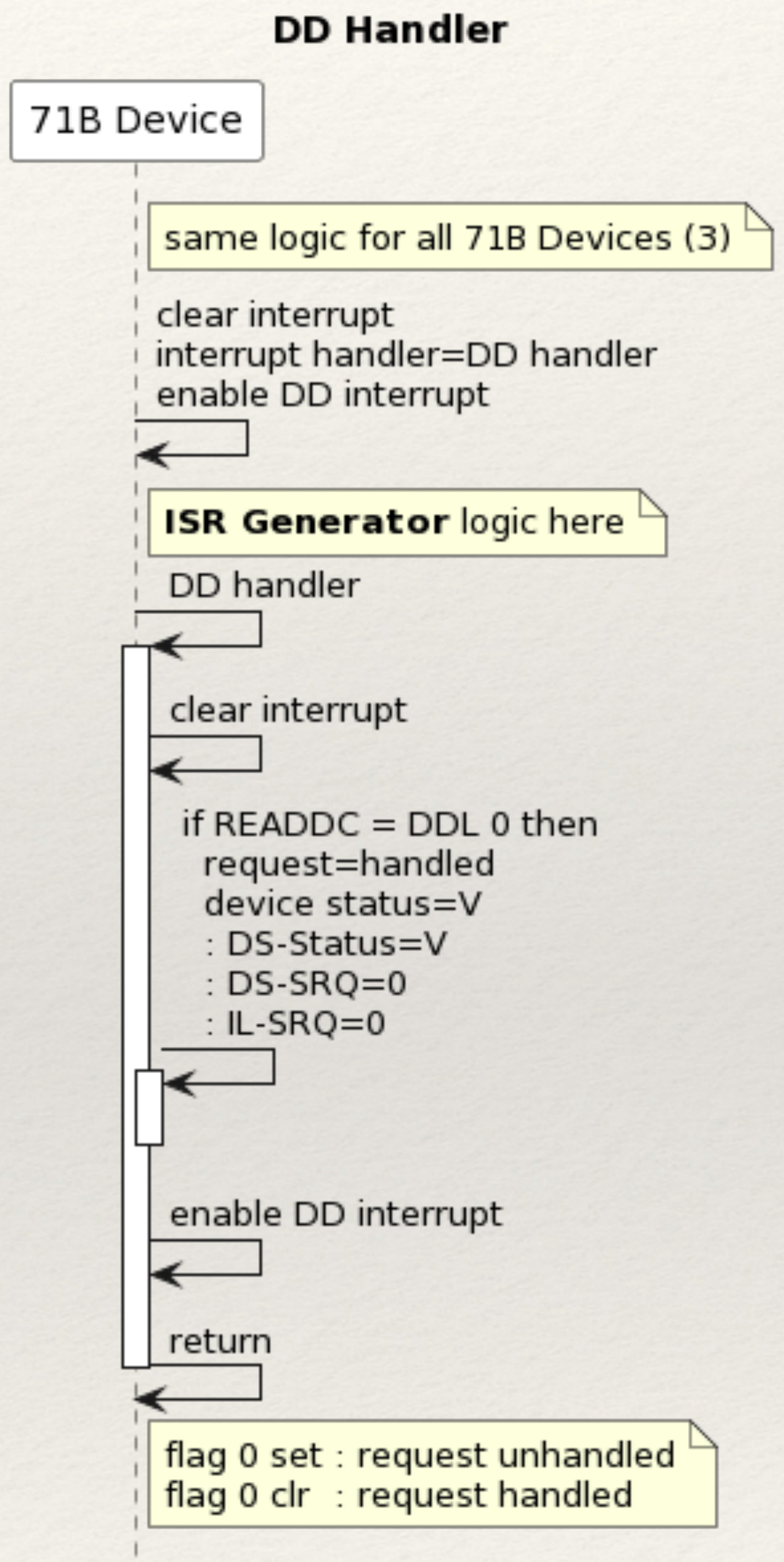












Bidirectional Communication

Multiple Devices - Parallel Polling - BC3D Implementation

Device

```
 5 ! BC3D                    Bidirectional comm. prog. 3, device side
10 DELAY 0,0                 Set display as fast as possible
 @ RANDOMIZE                 Random number generator
 @ A=MYADDR                  Save my loop address
 @ CFLAG 0,1                 Flag 0 (waiting for DDL 0) & flag 1 (waiting)

15 N=READINTR                Clear interrupt
 @ ON INTR GOSUB 50          Set interrupt handler
 @ ENABLE INTR 1             Enable interrupt on device dependent message

20 FOR V=1 TO 4              Loop to set or clear flags 1 to 4
25   DISP "D="&STR$(A)&"  S="&STR$(V) Ex.: "D=3  S=2" (Device & Status)
30   SFLAG 0                 Activate wait for DDL 0 message flag
 @   REQUEST 64+V            Raise SRQ bit while passing flag value
35   IF FLAG(0)=1 THEN       DDL 0 received ?
       SLEEP                 No: go to sleep
 @     GOTO 35               Infinite DDL 0 wait loop
40   SFLAG 1                 Activate wait flag
 @   WAIT IP(RND*10+1)/10    Wait 100 ms to 1 sec, asynchronous simulation
 @   CFLAG 1                 Deactivate wait flag
45 NEXT V
 @ GOTO 20                   Infinite loop
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

Device

```
50 N=READINTR                Enter interrupt handler, clear interrupt
 @ C=READDDC                 [00..31: DDT 0..31] & [32..63: DDL 0..31]
55 IF C=32 THEN              If DDL 0 received then
     CFLAG 0                   Clear DDL 0 wait flag
 @   REQUEST V                 Clear SRQ bit
60 ENABLE INTR 1             Re-enable device dependent interrupt
 @ RETURN                    Leave interrupt handler
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

Bidirectional Communication

Multiple Devices - Parallel Polling - BC5C Implementation

Controller

```
5 ! BC5C                       Bidirectional comm. prog. 5, controller side
10 OPTION BASE 0               Make array index start at zero
 @ DELAY 0,0                   Set display as fast as possible
 @ DIM D$[60]                  Space for 8 devices (6 chr/device + some room)
15 CONTROL ON                  Reconfigure loop
 @ N=NLOOP()                   Get the number of devices in the loop
20 DESTROY A1, A2              Delete devices address & devices status arrays
 @ SHORT A1(N)                 Create an address array for all devices
 @ A1(0)=0                     Initialize 71B found counter
25 FOR I=1 TO N                Scan loop for 71Bs
30   IF DEVAID(I)=3 THEN       Is device a 71B ?
       A1(0)=A1(0)+1           Yes, increment 71B found counter
 @     A1(A1(0))=I             Save 71b address in devices address array
35 NEXT I
40 IF A1(0)=0 THEN             Quit if no 71B has been found
     DISP "No 71B found"
 @   DESTROY A1, A2            Exit: delete arrays
 @   END                       End of execution …
45 SHORT A1(A1(0)),A2(A1(0))   Resize/create arrays to match 71Bs found
 @ S=0
50 GOSUB 85                    Display what we have
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

Controller

```
55 SEND CMD 21 UNL                              Parallel poll unconfigure (PPU, 15h, 21d)
 @ FOR I=1 TO A1(0)                             For all 71Bs
 @   SEND LISTEN A1(I) CMD 128+8+I-1 UNL        Enable parallel polling
 @ NEXT I                                       PPE 1:7..0 → 128 (PPE) + 8 (1:) + I-1 (7..0)
60 S=PPOLL()                                    Poll the loop to see If we have a service request
 @ IF S#0 THEN                                  Do we have a service request ?
     GOSUB 70                                   Yes: call SRQ handler
65 WAIT .15                                     Slowing polling loop to see multiple SRQs
 @ GOTO 60                                      Infinite loop polling
70 FOR I=1 TO A1(0)                             Enter SRQ handler, for each 71B found
75   IF BIT(S,I-1)=1 THEN                       Is the SRQ bit set for this device ?
       A2(I)=SPOLL(A1(I))                       Get 71B status and save it in status array
 @
         SEND UNT UNL LISTEN A1(I) DDL 0 UNL
80 NEXT I                                       ↑ Yes, notify device → SRQ handled
85 D$=STR$(A1(0))&":"&DTH$(S)[5,5]              71B found & SRQ received
90 FOR I=A1(0) TO 1 STEP -1                     In reversed order to match PP bits pattern
 @   D$=D$&CHR$(92)&STR$(A1(I))&":"&DTH$(A2(I))[4,5]
 @ NEXT I                                       ↑ Add 71B address then add 71B status
95 DISP D$                                      Ex.: 3:5\4:41\3:00\2:43
 @ RETURN                                       Leave SRQ handler,
                                                ↑ end sub for line 50 and 60
```

Default
Interrupt
Communication
Device Status
Device Address
Service Request

Bidirectional Communication

Demonstration Video

What's Next ?

- Create your own communication program.
- Define a set of DDL and DDT messages that make sense for what you are trying to do.
- For each DD message, define what and how the data will be sent.
- Implement the logic for each DD message: send, receive & handling.

- Ex.: multi-casting a string on request
  - Status: 5 = I have a string to multi-cast
  - Message format for message originator (DDT 0) and message receivers (DDL 2):
    byte 1 = string length
    following bytes = string itself
  - Logic
    1. Device #3 set status to 64+5
    2. Controller get the status
    3. Ctrl send a DDT 0 to device #3 to get the msg
    4. Ctrl send a DDL 2 w. msg to the other devices
    5. Each device get and display the message

Reference

Manuals

- IL Core Documentation
  - 5955-9425 The HP-IL System: An Introductory Guide (ISBN: 0-931988-77-2)
  - 82166-90016 The HP-IL Integrated Circuit User's Manual
  - 82166-90017 The HP-IL Interface Specification
  - 82166-90020 The HP-IL Interface Kit Technical Guide
  - 82166-90024 Application Note: Firmware Design for HP-IL Devices
- 71 Handheld Computer
  - 82401-90001 HP 71 HP-IL Interface Owner's Manual
  - 82402-90002 HP 71 Dual HP-IL Adapter Owner's Manual
  - 82402-90003 HP 71 Dual HP-IL Adapter Owner's Manual Addendum
  - 82401-90023 HP 71 HP-IL Module Internal Design Specification Volume I
- 75 Handheld Computer
  - 00075-90144 HP 75 RIOWIO Instruction Card
  - 00075-15001 HP 75 I/O ROM Programming Techniques Manual
  - 00075-90259 HP 75 I/O ROM Programming Techniques Manual Addendum

Virtual Floppy Disk

- NLOOPLEX Missing IL functions: SLEEP, NLOOP, SRQ & PPOLL
- XHPIL Missing IL functions: MYADDR, etc.
- UC1D Unidirectional Comm. Using Interrupt Handler, Device side
- UC2C Unidirectional Comm. Using Interrupt Handler, Ctrl side
- UC3C Unidirectional Comm. Using SRQ() Function, Ctrl side
- BC1D Bidirectional Comm. Using SRQ() & DD messages, Device side
- BC2C Bidirectional Comm. Using SRQ() & DD messages, Ctrl side
- BC3D Bidi. Comm. with Multiple Devices, Device side
- BC4C Bidi. Comm. with Multiple Devices, Serial Polling, Ctrl side
- BC5C Bidi. Comm. with Multiple Devices, Parallel Polling, Ctrl side
- PLOADER Copy the above LEX and BASIC files into your HP-71B

PLOADER

- The programs loader utility allow you to load LEX and BASIC files used in this presentation into an IRAM module of your HP-71B.
- PORT(0.00) and TAPE(1) are used, change lines 20 & 25 if you have other preferences.
- To load the programs, do the following:
  - CONTROL ON — *Take control of the loop*
  - COPY PLOADER:TAPE(1) — *Transfer program from tape into main memory*
  - FREE :PORT(0) — *Create a 4K IRAM module (0.00)*
  - RUN PLOADER — *Copy files from :TAPE(1) to :PORT(0.00)*
  - PURGE PLOADER — *Remove program from main memory*
  - CAT :PORT(0) — *Show 4K IRAM (0.00) module content*
- To unload the programs, do the following:
  - CLAIM :PORT(0) — *Destroy the 4K IRAM module (0.00)*

Questions ?

HP-IL

Thank You!

by Sylvain Côté